NEC

PRODUCT DESCRIPTION

EVACHIP

μPD 7500 G

NEC ELECTRONICS (EUROPE) GMBH

PRODUCT DESCRIPTION

EVACHIP

µPD 7500 G

4/83 V1.1

CHAPTER 1. GENERAL DESCRIPTION

The μPD7500 is specifically developed as an evaluation chip of μPD7500 Series Single Chip 4-Bit Microcomputer. It has 4-bit parallel processing Arithmetic Logic Unit, data memory (RAM), bus interface, I/O ports, 8-bit serial interface, programmable 8-bit timer/event counter, and vector interrupt functions all in a single chip.

The μPD7500 is capable of making direct access to the external program memory of up to 8 K bytes. It can also allow expansion of input/output ports using 8243 I/O expander. This microcomputer is also provided with the stand-by function for saving the power while making timer operation.

In the μPD7500, there are two types of selectable instruction sets, selection of which can be made according to user's needs.

Utilizing the flexible features such as above, the μPD7500 can be used as a general purpose microcomputer.

1.1 Features

(1) μPD7500 series evaluation chip.

(2) 4-bit microcomputer

(3) Two types of instruction set (SET A and SET B)

SET / Instruction Function	SET A (110 Kinds)	SET B (70 Kinds)
Transfer	35	17
Arithmetic operation	24	10
Rotation	2	2
Jump	4	4
Call, return	5	4
Port operation	12	10
Control	12	10
Skip	11	8
Others	5	5

(4) Instruction cycle : 10 μS/200 kHz

(5) External program memory capacity: 8192 words x 8 bits

(6) Internal data memory (RAM) capacity: 256 words x 4 bits

(7) Ample addressing function:
Register, register indirect, auto increment, auto decrement, immediate, and direct

(8) RAM stack

(9) Vectored interrupts:
External : 3 (INTO, INT1, INT2)
Internal : 2 (INTS, INTT)

(10) Programmable 8-bit timer/event counter

(11) 8-bit serial interface (operation possible with 3 types of serial shift clock)

(12) 8243 I/O expander interface

(13) 46 I/O lines

- 4-bit input port (P00 - P03) ... used common to $\overline{SCX}$ (P01), $\overline{SO}$ (P02), $\overline{SI}$ (P03)
- 4-bit I/O ports with output strobe (P10 - P13) ...
- Output latched 4-bit output port (P20 - P23) ... used common to $\overline{PSTB}$ (P20) and $\overline{PTOUT}$ (P21).
- Output latched 4-bit output port (P30 - P33)
- Output latched 4-bit I/O port (P40 - P43)
  Output latched 4-bit I/O port (P50 - P53)
  } Can be in unit of 8-bit.
- Output latched programmable 4-bit I/O port (P60 - P63)
- Output Latched 4-bit I/O port (P70 - P73)
- 14-bit address/control/data bus (BUS0 - BUS13)

(14) Power saving function in Stand-by (STOP/HALT) mode.

(15) On-chip RC Oscillator circuit for system clock (external drive possible).

(16) On-chip crystal oscillator circuit for count clock (external drive possible)

(17) Reference clock (LCD CL) output for external LCD control circuit

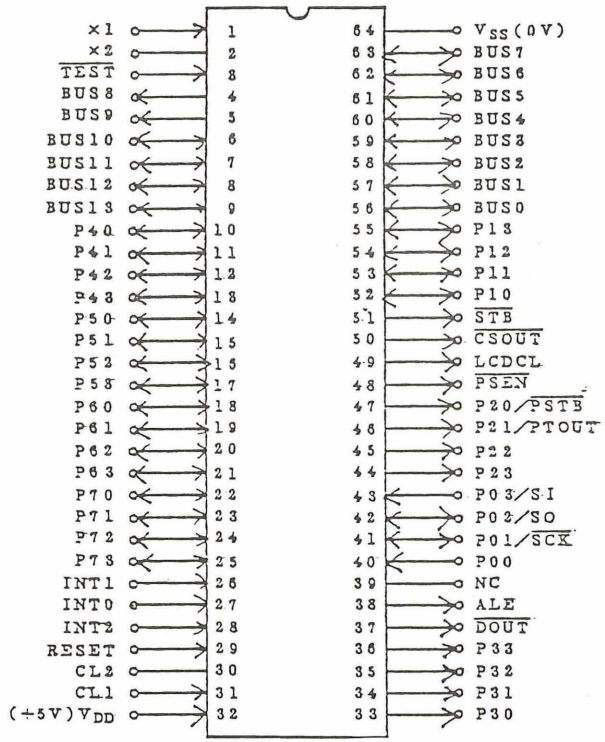
(18) CMOS

(19) Low power consumption

(20) Single power supply (+5 V)

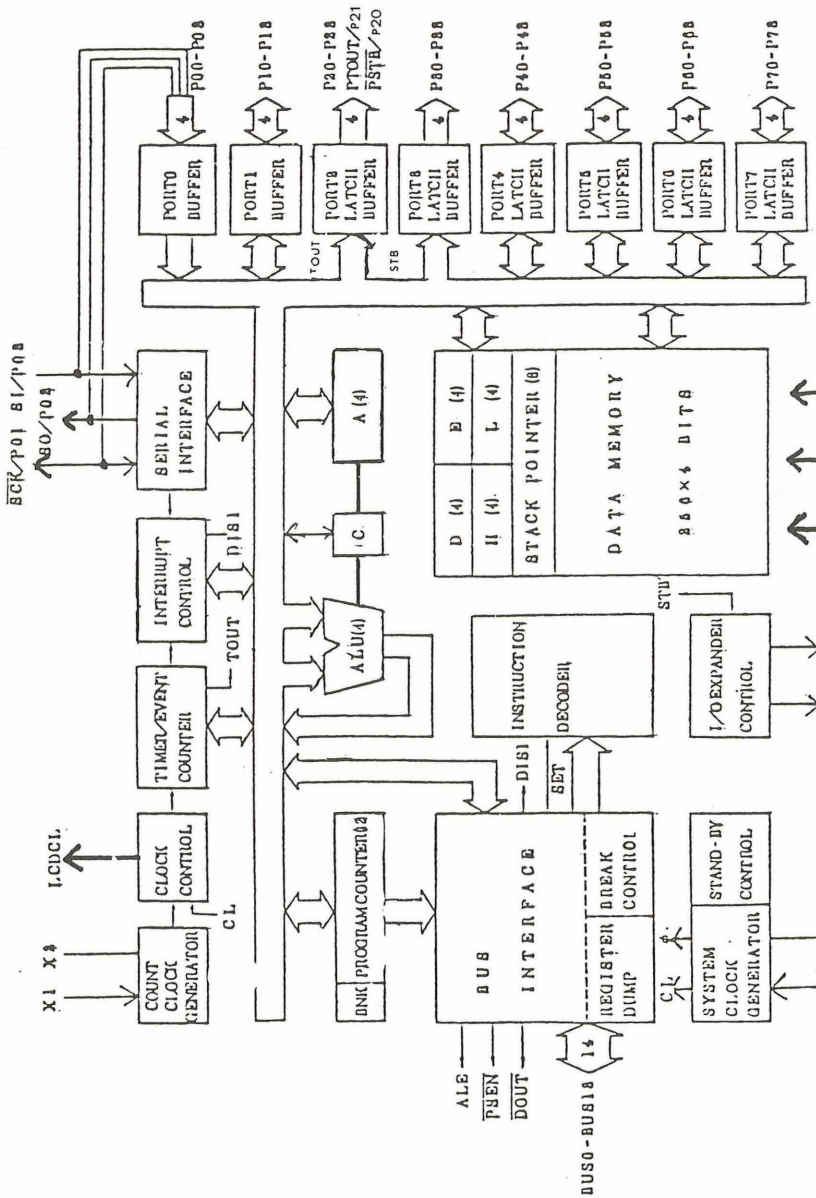
(21) 64 pin Quad-In-Line plastic package

1.2 Pin Configuration



X1,X2	:Crystal	INTO,1,2	:Interrupt	LCD CL	:LCD Clock
$\overline{\text{TEST}}$	:LSI Test	RESET	:Reset	$\overline{\text{CSOUT}}$	:Chip Select Out
Buso-13	:Bus	CL1,2	:Clock	$\overline{\text{STB}}$	:Strobe
P00-03	:Port0	$\overline{\text{DOUT}}$	:Data Out	NC	:No Connection
P10-13	:Port1	ALE	:Address Latch Enable		
P20-23	:Port2	$\overline{\text{SCK}}$	:Serial Clock		
P30-33	:Port3	SO	:Serial Out		
P40-43	:Port4	SI	:Serial In		
P50-53	:Port5	PTOUT	:Port Timer Out		
P60-63	:Port6	$\overline{\text{PSTB}}$	:Port Strobe		
P70-73	:Port7	$\overline{\text{PSEN}}$	:Program Store Enable		

1.3 Block Diagramm



CHAPTER 2. PIN DESCRIPTION

2.1 P00 - P03 (Port 0) ... 3 State Input/Output

The 4-bit input pins (P00 - P03) of port 0 (4-bit input port) functions not only as port pins, but also as serial clock input/output pin ($\overline{SCX}$/P01) serial data output pin (SO/PO2), and serial data input pin (SI/P03) for serial interface.

The operation mode of these input and output pins are decided by the contents of the shift mode register, but data can at any times be loaded in the accumulator over P00 - P03 lines by executing port input instructions (IP, IPL), including a case where these are set as serial interface input/output pins. The RESET input makes Port 0 in high impedance state.

2.2 P10 - P13 (Port 1) ... 3 State Input/Output

These are 4-bit input/output pins for port 1 (4-bit input/output port). At the time of execution of port input instructions (IP, IP1, IPL), data on the line is loaded in the accumulator, and at the time of execution of instructions (OP, OPL), the contents of accumulator are output. Port 1 does not equip an output latch, and during port output instruction execution, a strobe signal is automatically output from $\overline{\text{STB}}$ terminal (and $\overline{\text{PSTB}}$/P20 pin) so that external circuit can latch output data. These are suited for data output to memory and peripheral circuits requiring the strobe signal.

As another major function, P10 - P13 pins are used for interface with 8243 I/O expander. P10 - P13 pins are connected to P20 - P23 pins of 8243, and they are used for output of commands to 8243 and also for input and output of data during the execution of 8243 operating instructions (LP, IPL, OP, OPL, ANP, ORP, RPBL, and SPBL).

Port 1 is usually in a high impedance state, and the output is driven only when the instruction for operating the port1 is executed.

2.3 P20 - P23 (Port 2)... 3 State Output

These are 4-bit output terminals for port 2 (latched 4-bit output port). The contents of accumulator are latched to the output latch and output from P20 - P23 pins by execution of output instructions (OP, OPL).

The contents of the output latch are freely set and reset by port logic instructions (ANP, ORP), so output level can be controlled in units of bit.

P20 serves as an output terminal of the strobe signal ($\overline{\text{PSTB}}$/P20) at the time of port 1 data input and output, and P21 as the output terminal of the timer output flip-flop (TOUT) signal (PTOUT/P21).

P20 - P23 output pins are of 3 state type, and are set in a high impedance state by reset input. When instructions (OP, OPL, ANP, ORP) for operating the port 2 are executed, these outputs become active (not 3-state).

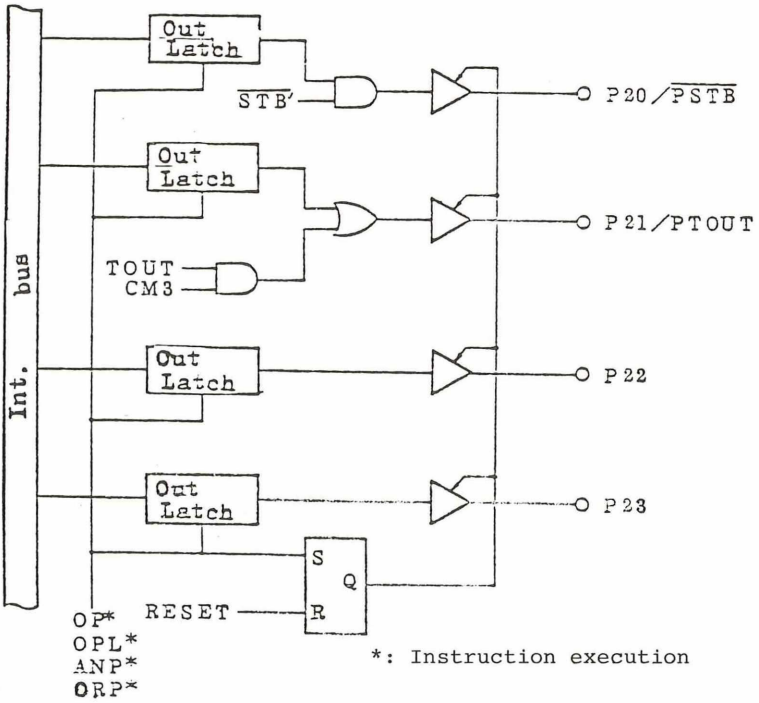
When using P20 and P21 as $\overline{\text{PSTB}}$ and PTOUT signal, therefore, it is necessary to execute in advance a port 2 operating instruction that outputs inactive levels of these signals.

The $\overline{\text{STB}}$ signal is normally at a high level. It is forced low only when a port 1 operating instruction that needs strobe signal. (Port 1 outputs and I/O Expander Operation)

The TOUT signal is a timer output flip-flop output signal. It is output to the PTOUT pin if bit 3 of the clock mode register (CM3) is set (1).

The contents of output latch once set are not affected until the port 2 operating instruction is next executed, or until RESET input is made. So, the level of each outputs pins are held, but with regard to P20 and P21 pins, a change may be made if they are being used as $\overline{\text{PSTB}}$ and PTOUT.

Fig. 2-1 Port 2 Configuration



Port 2 Operating Instruction

2.4 P30 - P33 (Port 3) ... 3 State Output

These are 4-bit output pins for port 3 (latched 4-bit output port). The contents of output latch can be set and reset in units of bit by the port logic instructions (ANP, ORP), so that P30 - P33 output signals can be controlled in units of bit.

The data once written in the output latch is held until another instruction for operating the port 3 is executed, or until RESET input is made.

These outputs are set in a high impedance state by reset input, and it becomes active when an instruction (OP, OPL, OP3, ANP, or ORP) for operating port 3 is executed.

When port 3 output instructions (OP, OP3, OPL) are executed, the content of accumulator is latched to the output latch and are output to P30 - P33 pins.

2.5 P40 - P43 (Port 4), P50 - P53 (Port 5) ... 3 State Input/Output

P40 - P43 are 4-bit input/output pins for port 4 (output latched 4-bit input/output port), and P50 - P53 are for (output latched 4-bit input/output port), the contents of each 4-bit line are loaded in the accumulator at the time of execution of port input instructions (IP, IPL), and the contents of accumulator are latched to the output latch during the execution of port output instructions (OP, OPL), and output to 4-bit line at the same time.

The contents of output latch are set and reset in units of bit by the port logic instructions (ANP, ORP), so that each output may be controlled in units of bit.

Port 5 and port 4 operate as a pair to permit the input of IP54 instruction in units of 8 bit, and the output of CP54 instruction in units of 8 bits with the accumulator and the data memory (addressed by pair register HL).

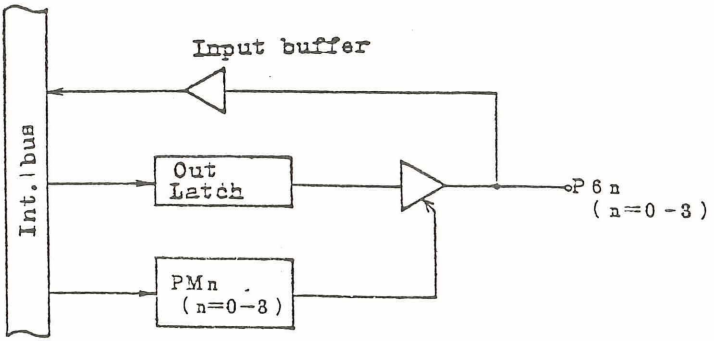
The data once written in the output latch is held until an instruction for operating the port 5 or port 4 Output latches next, or until the RESET input is made.

These pins are automatically placed in the input mode (the output buffers are in high impedance state) with the port input instruction (IP, IPL, or IP54) executed.

2.6 P60 - P63 (Port 6) ... 3 State Input/Output

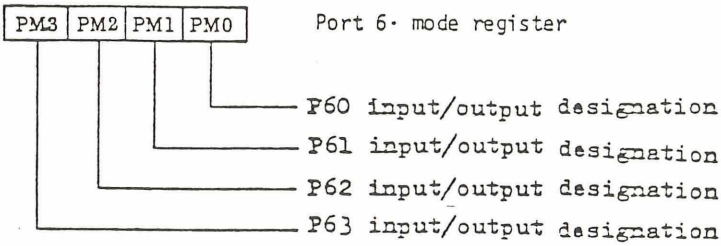
These are 4-bit input/output pins for port 6 (output latched progammable 4 bit input/output port). The port 6 can be set in the input or output mode in units of bit by the port 6 mode registers (PMO - PM3).

Fig. 2-2. Port 6n Configuration



Each bit of port 6 is set in the input mode when the corresponding bits of port 6 mode register are 0, and is set in the output mode when the bits are 1. The contents of the port 6 mode register are set with the contents of the accumulator transferred by OP, OPL instructions.

Fig. 2-3 Port 6 · Mode register



PMn	Contents Designated
0	P6n input mode (output buffer in a high impedance state)
1	P6n output mode (output buffer ON)

Port 6 is operated by port inputs (IP, IPL), outputs (OP, OPL), and port logic instructions (ANP, ORP). During the execution of input and output instructions, transfer of data is made in units of 4 bits with the accumulator.

The movement of port 6 and P60 - P63 pins during the execution of each instruction differs depending upon whether each bit is set in the input mode or output mode.

(1) When Set in the Input Mode (PMn = 0)

The output buffer is turned off and placed in a high impedance state. When the input instruction is executed, data on the P6n line is loaded in the corresponding bits of the accumulator.

Even in this case, the output latch is effective so that it can be set and reset by either an output instruction or port logic instruction. The contents of output latch are held until either the port 6 output instruction or an port logic instruction is next executed, or RESET input is made. The contents of the output latch being held are output to P6n pin via the output buffer when this bit is set in the output mode.
In the input mode, incidentally, it is not possible to load the contents of output latch in the accumulator.

Since all the bits of the port 6 mode register are initialized to '0' by RESET input. So, all the bits of port 6 are set in the input mode.

(2) When Set in the Output Mode (PMn = 1)

The output buffer is turned on, and the contents of the output latch are output to P6n pin via the output buffer. The contents of the output latch are set and reset by output instruction or port logic instruction. Also, the contents of the output latch are held until the port 6 output instruction or port logic instruction is executed or until RESET input is made. This also causes the corresponding output pins to be held.

The input buffer is effective even in the output mode. The data output to P6n pin can be loaded in the corresponding bits of the accumulator by an input instruction via its own outbut buffer. This is effective for monitoring the output data.

2.7 P70 - P73 (Port 7) ... 3 State Input/Output

These are 4-bit input/output pins for port 7 (output latched 4 bit input/output port). At the time of execution of port instructions (IP, IPL), the data on the P70 - P73 line is loaded in the accumulator. At the time of execution of port instruction (OP, OPL), the contents of the accumulator are latched to the output latch, and at the same time output to P70 - P73 pins.

The contents of output latch can be set and reset in units of 4-bit by port logic instructions (ANP, ORP), and each output signal can be controlled in units of bit.

The data once written in the output latch is held until an instruction for operating the port 7 is executed next, or until RESET input is made.

These can be automatically be placed in the input mode (the output buffer is high impedance state) by RESET input, or execution of port instructions (IP, IPL).

2.8 BUSO - BUS13 (Address/Data/Control Bus) ... 3 State Input/Output

These are multiplexing address/data/control buses. These are operated by being divided into four states (T1, T2, T3, T4) in a machine cycle. BUS8 and BUS9 are output only.

In the T1 state, bank flag (BNK) and program counter (PC11-0), which are for fetching an instruction code that is executed in the next machine cycle are output on the BUS0 to BUS13.

The instruction code, which is addressed by the BNK and PC11-0, is fetched in the state of T2. In T2, the break request signal, interrupt prohibit request signal, and instruction set select signal are further fetched.

In T3, bus operation varies depending upon whether the machine cycle is the normal operation cycle or the break cycle. If this is the operation cycle, both internal data memory reference address and its write data are output.
In the break cycle, the contents of H, L, D, E registers, accumulator, skip flag (SK1, SK0), carry flag (C), and stack pointer (SP) are output.

When the STOP, HALT instructions are executed, these bus signals are held in T1 state until the STOP or HALT mode are released. After release of STOP and HALT, bus operation is continued. T4 is an idle state, so that meaningful bus input and output operations are not made, and internal data processing is made.

2.9 ALE (Address Latch Enable) ... Output

This is a strobe signal for externally latching the program memory address to be output to BUSO - BUSB pins at T1 timing. It is fixed at a high level during the execution of STOP and HALT instructions.

2.10 $\overline{\text{PSEN}}$ (Program Store Enable) ... Output

This is a strobe signal for input of instruction code and control signal to BUSO - BUS13 at T2 timing. It is fixed at a high level during the execution of STOP and HALT instructions.

2.11 $\overline{\text{DOUT}}$ (Data Output) ... Output

This is a strobe signal for externally latching various information to be output on BUSO - BUS13 at T3 timing. It is fixed at a high level during the execution of STOP and HALT instructions.

2.12 INTO (Interrupt 0) ... Input

This is an external interrupt input terminal which is active at rising edge. It becomes effective when the INTO signal has changed to the high level from the low level. To allow an interrupt to be again acknowledged after once acknowledged it, it is necessary to once set the INTO input to the low level and then to raise it to the high level.

The INTO interrupt processing is common to INTS (serial interface interrupt), selection of which is carried out by the shift mode register.

INTO is used after releasing the stand-by mode (STOP and HALT modes).

2.13 INT1 (Interrupt 1) ... Input

This is an external interrupt input terminal. It's operation mode is determined by the contents of INT1 mode register. The following four modes can be selected.

(1) Rising egde active.
(2) Falling egde active.
(3) Both Rising/Falling egdes active.
(4) Ignored.

INT1 is effective even during the HALT mode. It is used for release of this mode, but it cannot be used for release of STOP mode. Even if INT1 input becomes active during the STOP mode, the interrupt process is not made. Nor the request flag (INT1 RQF) is set. (Nothing occurs, ignored.)

2.14 INT2 (Interrupt 2) ... Input

This is the external interrupt input which is active at rising edge. INT2 is effective even during the HALT mode. It is also used for release of this mode, but it cannot be utilized for release of the STOP mode. Even if INT2 input has become active, interrupt process is not made in the STOP mode. The request flag (INT2 RQF) cannot be set.

2.15 $\overline{\text{CSOUT}}$ (Chip Select Out) ... Output

When 8243 I/O expander is to be operated by the port operating instruction, this signal becomes active (low). It becomes an identification signal for externally notifying that 8243 are accessed or chip select signal for 8243 if only one 8243 is used.

2.16 $\overline{\text{STB}}$ (Strobe) ... Output

During the execution of port 1 output instruction, the write strobe signal is output, and is used as write control signal of the memory and I/O devices.

When 8243 I/O expander is selected by the port operating instruction, the output becomes the PROG signal of 8243 to control data read and write operations.

With the port 2 output buffer turned on, the same one as this signal is output also to $\overline{\text{PSTB}}$/P20 pin.

2.17 LCD CL (LCD Clock) ... Output

The clock is directly output from the prescaler 3 in the clock control circuit. When externally providing an LCD circuit, this output is used as an operating clock for LCD.

The LCD CL does not stop neither in the STOP mode (When the clock source is a count clock system) nor in the HALT mode.

2.18 X1, X2 (Crystals)

These are crystal connection terminals for crystal oscillator circuit of the count clock generator circuit.

X1 becomes an external count clock input also. At this time, it operates also as an event pulse input. When an external clock is input to X1, X2 should be kept open.

2.19 CL1, CL2 (Clocks)

These are connection pins for resistor R and capacitor C of the RC oscillator circuit in the clock generator circuit.

CL1 also becomes an external clock input. At that time, it is necessary to make CL2 open.

2.20 RESET (Reset) ... Input

This is a high level active CPU reset input. It operates priority to all functions.

2.21 $\overline{\text{TEST}}$ (IC Test) ...

This is a test input pin for inspection of μPD7500. It should be fixed at a high level.

2.22 V_{DD}

This is a power supply pin.

2.23 V_{SS}

This is a GND potential.

CHAPTER 3. INTERNAL BLOCK FUNCTION

3.1 Bank Flag (BNK) ... 1 Bit

The bank flag designates the most significant bit (bank designation) for program memory address. With 0, bank 0 (OOOOH - OFFFH) is designated, and with 1, bank 1 (1OOOH - 1FFFH) is designated. At T1 timing for each machine cycle, it is output to BUS12 pin.

With the JMPL instruction, immediate data is loaded and set to 0 or 1.

At the time of execution of a call instruction (CALL, CAL, or CALT), and also at the time of an interrupt execution, the contents of the BNK are stored in the stack area, and it is cleared (0).

During the execution of a return instruction (RT, RTS, or RTPSW), the contents of the stack area are restored in BNK.

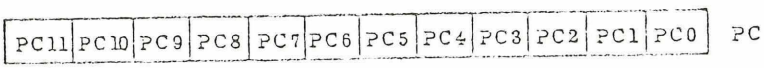
At the time of reset input, it is cleared.

The contents of the bank flag remain unaffected by increment or decrement of the program counter.

3.2 Program Counter (PC) ... 12 Bits

This is a 12 bit binary counter for holding the low-order 12 bit address information of the program memory.

Fig. 3.1 Program Counter Structure



The program counter is automatically incremented according to the number of bytes of the instruction whenever an instruction is executed.

At the time of execution of a call instruction (CALL, CAL, or CALT), or at the time of interrupt execution, the contents of PC at that time (already incremented for designating the next instruction, *i.e.* the return address) are stored in the stack memory and then a proper address is loaded to the program counter.

At the time of execution of a return instruction (RT, RTS, or RTPSW), the contents of stack memory are loaded in the PC.

The contents of PC are output to BUS11 - BUS0 pins at the T1 of each machine cycle.

3.3 Stack Pointer (SP) ... 8 Bits

This is a 8 bit register (SP7 - SP0) which stores the stack top address in case data memory is used as a LIFO stack.

This is decremented at the time of execution of a call instruction (CALL, CAL, or CALT), or a push instruction (PSHDE, PSHHL), and at the time of interrupt generation. And it is incremented at the time of execution of return instruction (RT, RTS, or RTPSW), or a pop instruction (POPDE, or POPHL).

To determine the stack area, SP is initialized by the TAMSP instruction. But at the time of execution of TAMSP instruction, 0 is unconditionally loaded to SP0. Incidentally, to set the most significant address of the stack area as FFH, the initial value of SP should be set at OOH.

The contents of SP can be read out at any time by TSPAM instruction, but the contents of SPO cannot be read at the time of execution of the TSPAM instruction, and 0 is unconditionally stored in bit 0 of data memory.

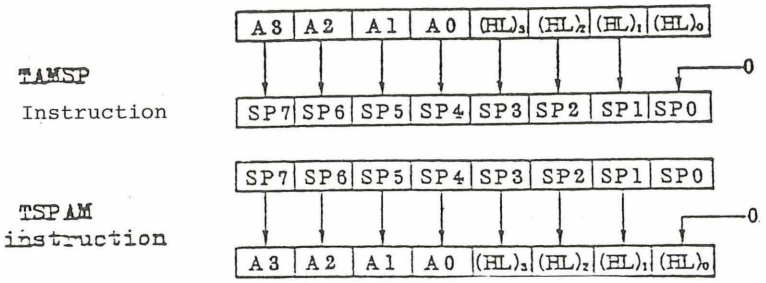


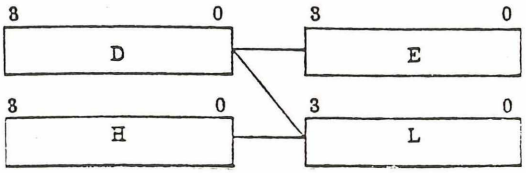
Fig.3.2 Movement of Data At Time of Execution of TAMSP Instruction and TSP AM Instruction

3.4 General-purpose Registers ... 4 x 4 Bits

There are four general-purpose 4-bit registers (D, E, H, L), each of which is operated in units of 4 bits, and DE, DL, and HL form an 8 bit pair register (D or H is the most significant and E or L is the least significant) to be used as a data pointer.

When a pair register HL functions as a data pointer, it performs the functions of automatic increment (+1) and automatic decrement (-1) for the L register. It shortens the program step and process time.

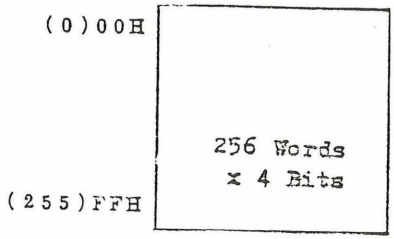
Fig. 3.3 General-purpose Register Configuration



3.5 Data Memory (RAM) ... 256 Words x 4 Bits

The data memory is a static RAM of 256 words x 4 bits in configuration and is used for storing process data. Also, it functions as a pair with the accumulator to make data processing in units of 8 bits.

Fig. 3.4 Data Memory Map

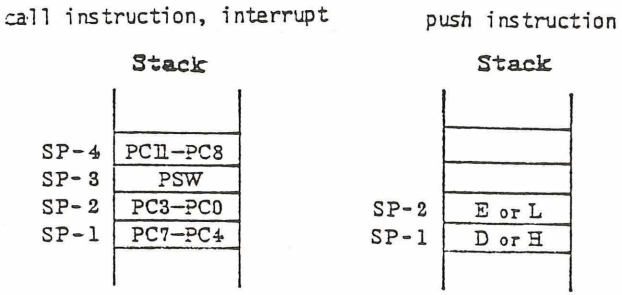


There are three types of data memory addressing.

- Direct: Direct address designation is made on the 2nd byte of instruction.
- Register indirect: Address designation can be made indirectly by the contents of pair register designated by the instruction.
- Stack: Indirect address designation is made by the contents of stack pointer (SP).

The desired area of the data memory can be used as a stack area. It is used as a stack area at the time of execution of a call or return instruction (CALL, CAL, CALT, RT, RTS, or RTPSW), and a push/pop instruction (PSHDE, PSHHL, POPDE, or POPHL), and at the time of interrupt generation. It is addressed by the stack pointer.

At the time of execution of a call instruction and interrupt generation, the contents of PC, and PSW are stored in the stack area, and at the time of execution of a push instruction, the contents of DE or HL are stored in the stack. The way these are stored are shown below.
The contents of data memory can be retained in the STOP mode even power supply is low voltage.



3.6 Accumulator (A) ... 4 Bits

The accumulator is a 4 bit register, and various arithmetic/logical operation is done mainly by the accumulator.

Operating as a pair with the data memory addressed by the pair register HL, processing may be done in units of 8 bit (higher-order bits for accumulator and lower-order bits for data memory).

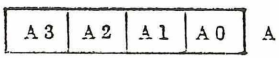


Fig. 3.5 Accumulator Configuration

3.7 Arithmetic Logic Unit (ALU) ... 4 Bits

The ALU is an arithmetic logic circuit of 4 bit configuration. With this unit, such processes as binary addition, arithmetic/logia/operation, comparison, and rotation are performed.

3.8 Program Status Word (PSW) ... 4 Bits

The program status word consists of skip flags (SK1, SK0), a blank flag (BNK), and a carry flag (C).

Fig. 3.6 Structure of Program Status Word

(1) Skip Flags (SK1, SK0)

The skip flag is used for memorizing the skip states indicated below.

- o String effect of LAI instruction.
- o String effect of LHLI and LHLT instruction.
- o Skip condition established by instructions other than string effect.

The skip flag is automatically set and reset by executing an instruction.

(2) Bank Flag (BNK)

The bank flag performs a switchover of program memory between bank 0 and bank 1. For further details, refer to section 3.1.

(3) Carry Flag (C)

At the time of execution of an additional instruction (ACSC) if a carry is generated from bit 3 of the ALU, the carry flag is set (1), but it is reset (0) if a carry is not generated. The carry flag is set (1) by the SC instruction, and reset (0) by the RC instruction. Also, its contents are tested by the SKC instruction. A data shift is made with the accumulator by a rotation instruction. (RAR or RAL).

The contents of PSW are automatically stored in the stack area at the time of execution of a call instruction (CALL, CAL, or CALT), and at the time of interrupt execution, and restored by RTPSW instruction. But BNK can be restored by RT or RTS instruction.

At the time of RESET input, SK1, SK0, BNK are cleared (0), and C is made uncertain.

3.9 System Clock Generator Circuit

The system clock generator circuit consists of an RC oscillator circuit and a 1/2 frequency divider circuit. The RC oscillator circuit oscillates a clock with a combination of externally provided resistor R and capacitor C, connected to CL1, CL2 pins. An external clock may be input from CL1. In this case, the RC oscillator circuit operates only as an inverting buffer.

The RC oscillator circuit output (CL) is sent to the clock control circuit and divided to be a count pulse (CP) of the timer/event counter. The CL frequency is equal to RC oscillator frequency or CL1 input clock frequency. The output of the RC oscillator circuit is frequency divided into 1/2, to become a system clock (ϕ) and sent to the CPU and serial interface.

Fig. 3.7a System Clock Generator Circuit Based on RC Oscillation

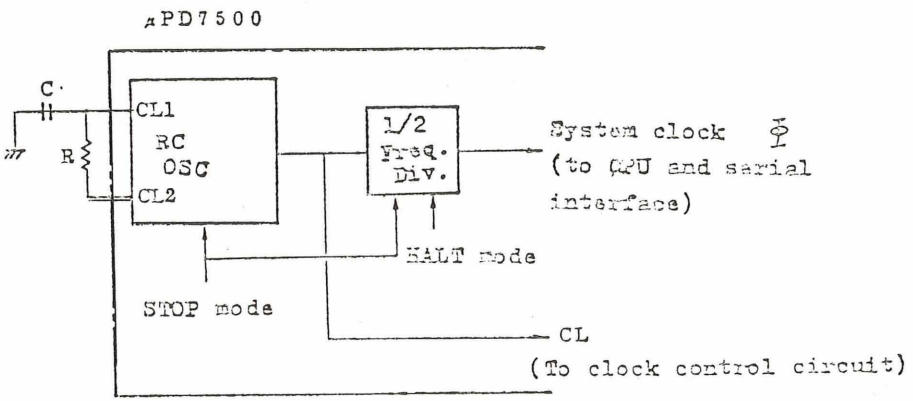
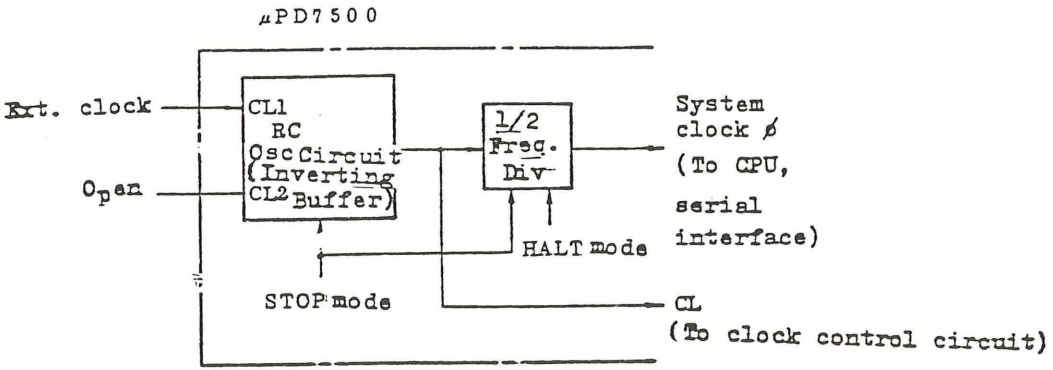


fig. 3-7b The system clock generator circuit by Ext, clock



The system clock generator circuit operates according to the stand-by (STOP/HALT) mode.

In the case of RC oscillation, the RC oscillator circuit and the 1/2 frequency divider circuit are stopped in the STOP mode (at the time of execution of the STOP instruction), thereby stopping the supply of CL and ϕ. In the HALT mode (at the time of execution of HALT instruction), only the 1/2 frequency divider is stopped. So, ϕ stops, but the CL continues to be supplied.

In the case of an external clock, CL1 input clock becomes CL via an inverting buffer so that even in the STOP mode, the supply of CL does not stop. In both the STOP mode and in the HALT mode, the circuit performs the function of stopping the 1/2 frequency divider circuit. In both modes, the supply of ϕ is stopped.

3.10 Count Clock Generator Circuit

The count clock generator circuit consists of a crystal oscillator circuit. The crystal oscillator circuit oscillates a clock with the crystal connected to X1, X2 pins, resistor R, and capacitors C. With this circuit, an external clock can be input from X1 input without making crystal oscillation. In this case, the crystal oscillator circuit merely functions as an inverting buffer.

The count clock generator output X is sent to the clock control circuit. The frequency of X is equal to the crystal oscillator frequency or X1 external clock.

The count clock generator circuit remains unaffected by the stand-by (STOP/HALT) mode.

And there, divided or not divided to be a count pulse (CP) of timer/event counter.

Fig. 3.8a Count Clock Generator Circuit Based on Crystal Oscillation

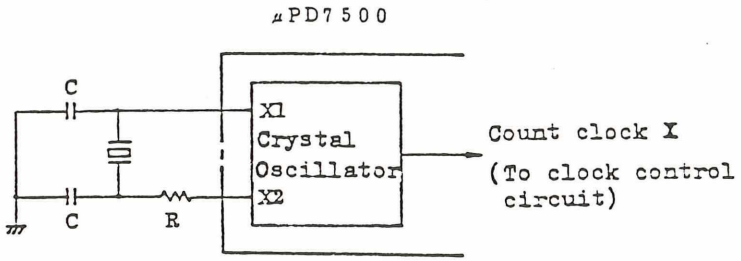
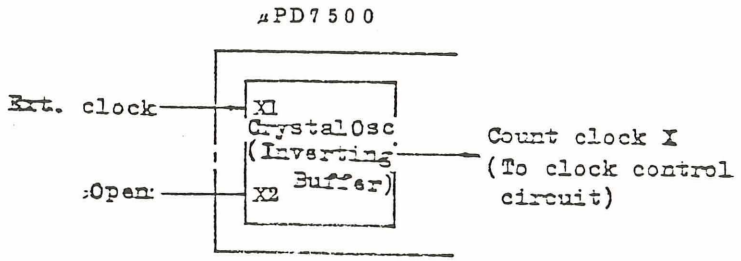


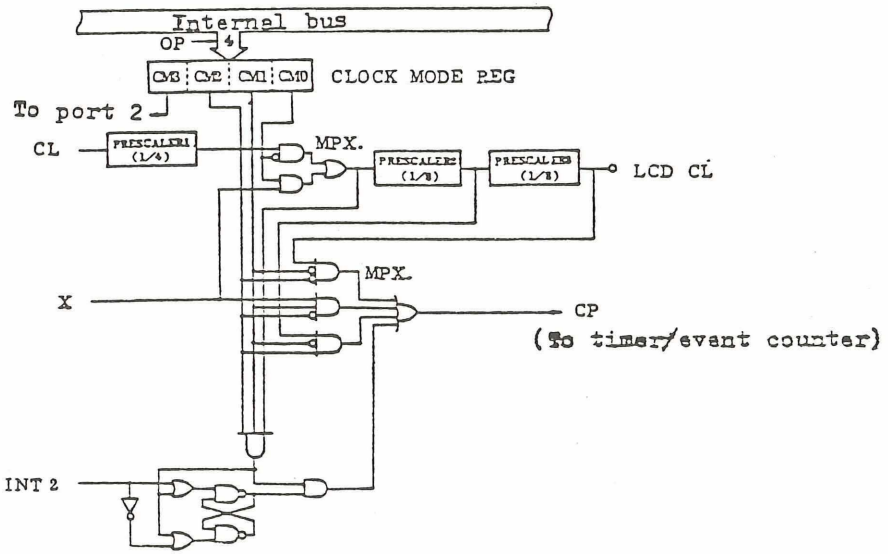
Fig. 3.8b Count Clock Generator Circuit Based on External Clock



3.11 Clock Control Circuit

The clock control circuit consists of 4 bit clock mode register (CM3 - CM0), prescalers 1, 2 and 3, and multiplexers. System clock generator circuit output (CL) and count clock generator circuit output (X), are input for source clock and the clock mode register selects clock sources and prescalers to supply a count pulse (CP) to the timer/event counter. Also, the output of prescaler 3 is output directly from LCD CL pin.

Fig. 3.9 Configuration of Clock Control Circuit



To the clock mode register, code is set by OP12, OPL (L = 12) instruction. CM3 is used for designation of timer out signal output, and with CM3 = 1, the output of the timer out F/F (TOUT) is output from PTOUT (P21) pin.

Fig. 3-10 Format of clock mode register

CM3	CM2	CM1	CM0

Clock mode register

CM2	CM1	CM0	Count Pulse Freq.	Output Freq. of LCD CL
0	0	0	$CL \times \frac{1}{256}$	$CL \times \frac{1}{256}$
0	0	1	$X \times \frac{1}{64}$	$X \times \frac{1}{64}$
0	1	0	X	$CL \times \frac{1}{256}$
0	1	1	X	$X \times \frac{1}{64}$
1	0	0	$CL \times \frac{1}{32}$	$CL \times \frac{1}{256}$
1	0	1	$X \times \frac{1}{8}$	$X \times \frac{1}{64}$
1	1	0	$CL \times \frac{1}{4} \wedge \overline{INT2}$	$CL \times \frac{1}{256}$
1	1	1	$X \wedge \overline{INT2}$	$X \times \frac{1}{64}$

CM3	Designation of Timer Out F/F Output
0	Stop
1	Output

When clock mode register CM2 = CM1 = 1, the count pulse (CP) to be supplied to the timer/event counter, timer/event counter is a signal which is 1/4 frequency divided clock of system clock output (CMO =0) i.e. CL x $\frac{1}{4}$, or not divided clock of count clock output (CMO=1) i.e. X, ANDed with the inverted level of INT2 signal.

In other words, only while the INT2 signal is at a low level, the count pulse is supplied to the timer/event counter. When the INT2 signal goes high from low-level, the count pulse is stopped, and immediately, the INT2 interrupt process can be initiated (with the count value thus derived). In interrupt service routine, it is possible to know the count value while INT2 was at low level.

3.12 Timer/Event Counter

The timer/event counter consists of an 8 bit count register, an 8 bit modulo register, an 8 bit comparator, and a timer out flip-flop.

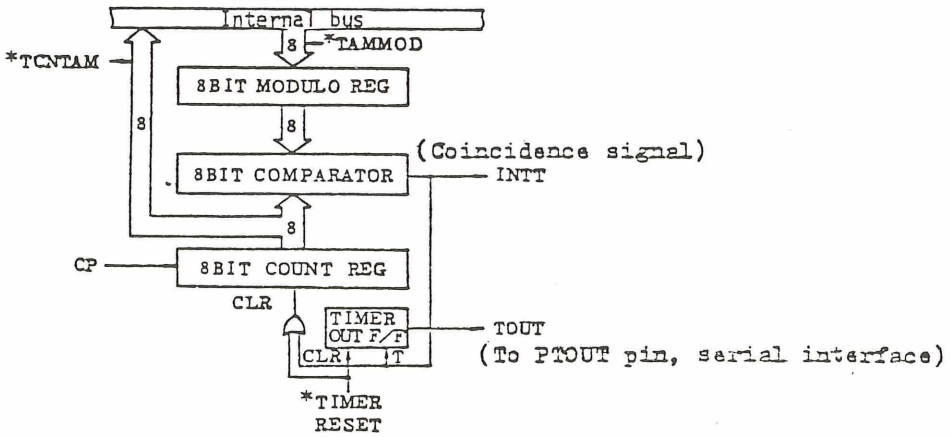
The 8 bit count register is a binary 8 bit up counter. It is incremented (+1) whenever a count pulse (CP) is input. At the time execution of the timer instruction, at the time of reset input, and at the time of coincidence signal generation, it is cleared to "0".

The 8 bit modulo register determines the number of counts of the count register with the contents set by the TAMMOD instruction. At the Time of RESET input, it is set to FFH.

The 8 bit comparator compares the contents of the count register with those of the modulo register and outputs the coincidence signal and timer interrupt (INT T) if these are equal.

The timer out flip-flop is flipped whenever a coincidence signal is generated. This timer out flip-flop output (TOUT) can also be output from PTOUT (P21) pin if bit 3 (CM3) of the clock mode register are set (1).

The TOUT signal is also used as a serial clock.



Notes:

1) CP is a count pulse selected by the clock mode register.
2) CM3 is an output designating bit of the timer out flip-flop from the clock mode register.
2) * denotes that the instruction is executed.

Next, timer operation and event counter operation are explained below.

(i) Timer Operation

Timer operation is initiated by clearing the contents of the count register by the TIMER instruction after setting the number of counts to the modulo register by the TAMMOD instruction using the count pulse (CP) as a time reference clock. Thereafter, the count register is incremented (+1) by being synchronized with the count pulse (in the case of an external clock, it is synchronized with the rising edge of CL1 input or X1 input). When the modulo register value coincides with the contents of the count register, a coincidence signal (INT T) is generated. After setting the interrupt request flag, the count register is cleared, and again count operation is repeated. Therefore, it operates as an interval timer whose interval is determined by the modulo register.

The count pulse is input into the count register at all times irrespectively of instructions, and to update the count value when the contents of the count register coincide with the modulo register the INT T request flag can be set. So, when a timer is not used, it is necessary to have the INT T interrupt prohibited in advance.

(ii) Event Counter Operation

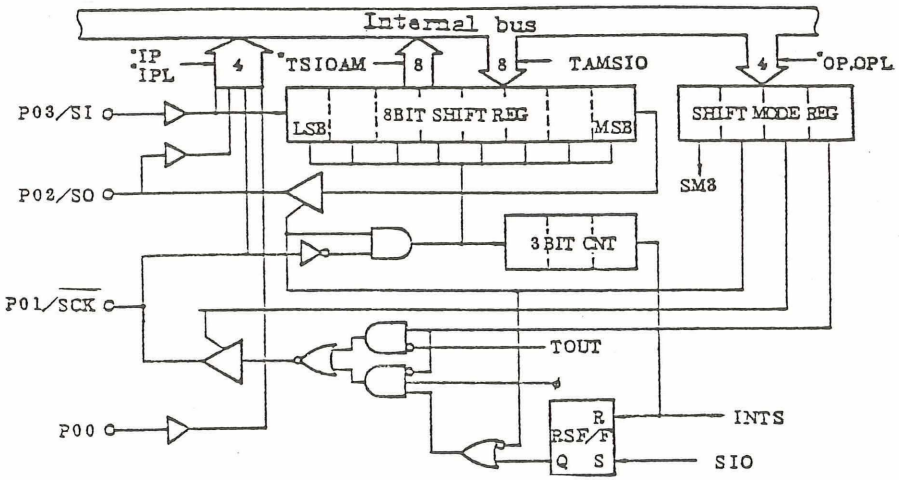
To carry out the operation of an event counter, the external event pulse should be input into the count clock generator circuit (X1) and the count clock (X) system should be selected as a count pulse (CP) by the clock control circuit.

By counting the external event pulse to be input from X1 input directly or being divided, the event counter operation is made to generate an interrupt based on the counts (events) determined by the value set to the modulo register. Incidentally, the count can be read out by the TCNTAM instruction.

Note: The contents of modulo register should be set the number less than 1 of desired event count but when this value is 0, the count register is held to 0 (clearing is done at the same time as coincidence detection). So, count operation is not possible. A value other than 0 should be set to the modulo register.

3.13 Serial Interface

The serial interface mainly consists of an 8 bit shift register, 4 bit shift mode register (SM0 - SM3), and a 3 bit counter. It is used for serial data input and output applications.

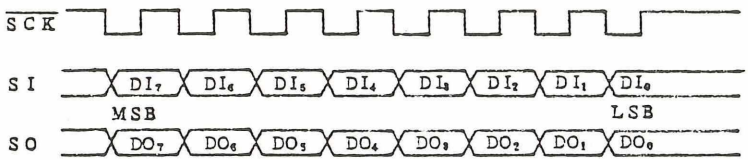


Notes:

(1) ø is an internal clock signal (system clock).
(2) TOUT is a timer out F/F output signal.
(3) * denotes that the instruction is executed.
(4) SM3 is to the interrupt control circuit.

The input and output of serial data is controlled by the serial clock. At the falling edge of the serial clock ($\overline{SCK}$ pin signal) the most significant bit (bit 7) is output from SO line, and at the rising edge the contents of the shift register are shifted 1 bit (bit n - bit n + 1), and at the same time data on the SI line is loaded in the least significant bit (bit 0) of the shift register.

The 3 bit counter (octal counter) counts serial clocks, and an internal interrupt signal INT S is generated whenever 8 shots are counted (the end of 1 byte serial data transfer), and the interrupt request flag (INT O/S RQF) is set.



Notes: DI : Serial data input
DO : Serial data output

When transmitting serial data, transmit data is set to the shift register by the TAMSIO instruction, and then that transfer is started by the SIO instruction. The end of 1 byte transfer can be determined by generation of an internal interrupt (INT S).

When receiving serial data, data reception is initiated by the SIO instruction, and after generation of an internal interrupt (INT S) upon the end of 1 byte transfer, the TSIOAM instruction should be executed to take receive data from the shift register.

Incidentally, the end of 1 byte transfer can be confirmed by testing INT S RQF with the SKI instruction, instead of making the interrupt processing.

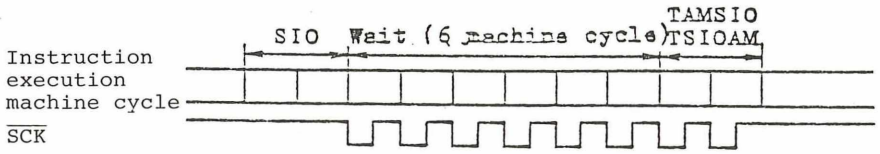
As serial clock sources, there are three types available - system clock ϕ, external clock ($\overline{SCK}$ input), timer out F/F output signal (TOUT). Selection of these can be made by bits 2 - 0 of the shift mode register (SMZ - 0).

In case where ϕ is selected, ϕ begins to be supplied to the serial interface as a serial clock by the SIO instruction, for control of serial data input and output. Also, it is output from the $\overline{SCK}$ pin.

When 8 shots of ϕ are supplied, its supply to the serial interface is automatically stopped, so that $\overline{SCK}$ output is held at the high level. After transfer of 1 byte, therefore, the input and output of serial data is automatically stopped. It is not necessary for the programmer to make control of the serial clock. In this case, the transfer speed is dictated by the system clock frequency.

In this mode, it is possible to read received data from the shift register (TSIONAM instruction executed) or to write the next transmit data (TAMSIO instruction executed) to the shift register, merely with waiting a period of 6 machine cycles after execution of the SIO instruction on the program. There is no need to wait for an internal interrupt (INT S) generation.

Fig. 3.14 Timing for Execution of TAMSIO, TSIOAM Instructions



When an external clock ($\overline{SCK}$ input) is selected, a serial clock is input from the $\overline{SCK}$ input. When eight shots of external serial clock are input, internal interrupt INT S is generated, to notify the end of 1 byte transfer. But internally, no serial clock prohibit control is made, it is necessary to input it by counting eight shots on the external clock source side. The transfer speed is dictated by the external serial clock, and it can be operated in a value range from DC to the maximum value limited by the electrical specification.

When the TOUT signal is selected, the coincidence signal of the timer/event counter 1/2 frequency divided becomes a serial clock for control of serial data input and output, and also it is output from the $\overline{SCK}$ pin. The transfer speed is determined by the count pulse supplied to the timer/event counter, and the value set to the modulo register. In the case of TOUT signal, the end of 1 byte transfer can be known by INT S generation, just as in the case of an external clock, but the TOUT signal is not prohibited so that it is necessary to control (stopping by count operation at every 16 interval) by the program.

When using an external clock and TOUT signal, execution of instructions such as SIO, TAMSIO, and TSIOAM should be made while the serial clock ($\overline{SCK}$) is at the high level. Operation is not guaranteed if instructions are executed in the course of $\overline{SCK}$ rise or fall, or while it is at the low level.

Fig. 3 - 15 Format or shift mode register

Shift mode register

Serial interface operation, and port 0 mode setting accompanying it.

SM2	SM1	SM0	Port 0 Function			Serial Operation
			PO3	PO2	PO1	
0	0	0	High Impedance	High Impedance	High Impedance	Stop
0	1	0	High Impedance	High Impedance	System clock continuously output	Stop
0	1	1	High Impedance	High Impedance	Timer out F/F signal output	Stop
1	0	0	S1 input	S0 output	Ext. shift clock input ($\overline{SCK}$)	Operation on external clock
1	1	0	S1 input	S0 output	System clock 8 clocks output ($\overline{SCK}$)	Operation on system clock
1	1	1	S1 input	S0 output	Timer out F/F signal output ($\overline{SCK}$)	Operated by timer out F/F

INT O or INT s selecti0n

SM3	Interrupt source
0	INTS
1	INT0

If a system does not require serial transfer, 8 bit shift register can be used as merely a register with serial operation stopped. Data read and write is possible by TSIOAM or TAMSIO instruction.

3.14 Interrupt Control Block

The μPD7500 has three types of external interrupts and two types of internal interrupts. But external interrupt INT 0 is common to internal interrupt INT S (in interrupt address and priority). The selection of interrupt sources, INT 0 or INT s is determined by the program.

Interrupt	Int./Ext.	Priority	Interrupt Address
INT T (coincidence signal from timer/event counter)	Int.	1	16
INT 0 (Interrupt signal from INT 0 pin)	Ext.	2	32
INT S (Shift and signal from serial interface)	Int.		
INT 1 (Interrupt signal from INT 1 pin)	Ext.	3	48
INT 2 (Interrupt signal from INT 2 pin)	Ext.	4	64

When an interrupt is generated and the corresponding interrupt request flag is set, the interrupt execution is initiated if the interrupt enable state (in which the interrupt master enable F/F is set, and the corresponding bit of interrupt enable register is also set) is now established. However, if a plurality of interrupts occur at the same time, the respective corresponding request flags are all set. But it is the one highest in priority which goes into interrupt execution and the interrupts lowest in priority are held.

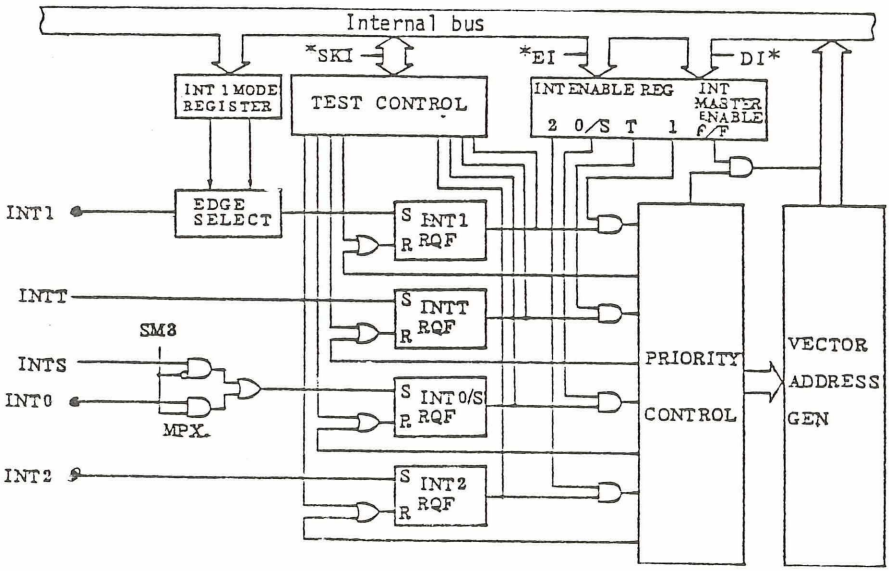
- Interrupt Execution -

(1) Master enable flip-flop is reset and all interrupts are in prohibit state.

(2) The contents of program counter (PC) and program status word (PSW) are stored in the stack.

(3) An interrupt start address is generated and a jump to the interrupt start address is made. The interrupt request flag that has generated an interrupt is reset at the same time.

Fig. 3.16 Interrupt Control Circuit Block Diagram



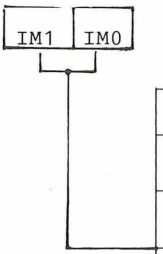
Notes:

(1) * denotes that the instruction is executed.

(2) SM3 is bit 3 of the shift mode register (selection of INT 0 or INT S).

3.14.1 Selection of INT1 Interrupt Signal Condition

INT1 has four types of interrupt signal conditions, and the selection of a desired condition is made by the INT1 mode register (1M1,0). Data setting to the mode register is made by OP13, OPL (L = 13) instruction.



INT1 Mode Register

1M1	1M0	Interrupt Signal Condition of INT1
0	0	Rising edge active.
0	1	Rising/falling edge active.
1	0	Any interrupt signals from INT1 pin are nulled. So, no operations such as INT1 RQF set, interrupt execute, and stand-by mode release can be made.
1	1	Falling edge active.

3.14.2 Selection INT 0/S

Selection of INT 0 or INT S is made in bit 3 of the shift mode register (SM3).

SM3	Selection of INT S, INT 0
0	INT S
1	INT 0

3.14.3 Interrupt Enable Register (IEO - IE3)

The interrupt enable register individually designate the enable or disable to the interrupt request for INT T, INT O/S, INT 1, and INT 2. If the corresponding bit of each enable register is 1, the interrupt is enabled, and if it is 0, the interrupt is disabled.

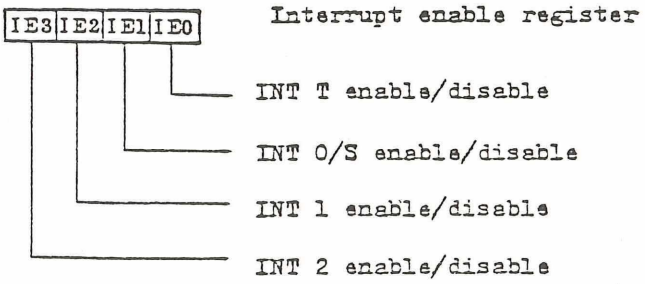


Fig. 3.18 Format of Interrupt Enable Register

The contents of interrupt enable register are set by EI, DI instructions.

At the time of EI instruction execution, the interrupt enable register is ORed with immediate data (= 0), and the result is stored in the interrupt enable register.

At the time of DI instruction execution, the interrupt enable register is ANDED with the inverted data of immediate data ($\neq$ 0), and the result is stored in the interrupt enable register. In these cases, interrupt master enable flip-flop is not affected.

4.4 Interrupt Master Enable Flip-Flop (IME)

The interrupt master enable flip-flop controls the enable or disable of the execution of all interrupts such as INT T, INTO/S, INT 1, and INT 2. After execution of an interrupt, it is reset, to disable the execution of a next generated interrupt.

The interrupt master enable flip-flop is set by the EI 0 instruction to establish an interrupt enable state, and is reset by the DI 0 instruction to establish an interrupt disable state. In these cases, no adverse effects are given to the interrupt enable register (IE3 - IE0).

CHAPTER 4. PROGRAM MEMORY STRUCTURE

The program memory can be externally provided up to 8 K bytes (0000H - 1FFFH address), and addressing is made by address information output on BUS12 - BUS0 pins. Normally, the content of the bank flag (BNK) is output on BUS12, and those of the program counter (PC11 - 0) on BUS11 - 0.

The program memory map is shown in Fig. 4-1.
The use of area of the program memory varies according to the instruction set. A difference takes place in the jump instruction, call instruction, and table reference instruction.

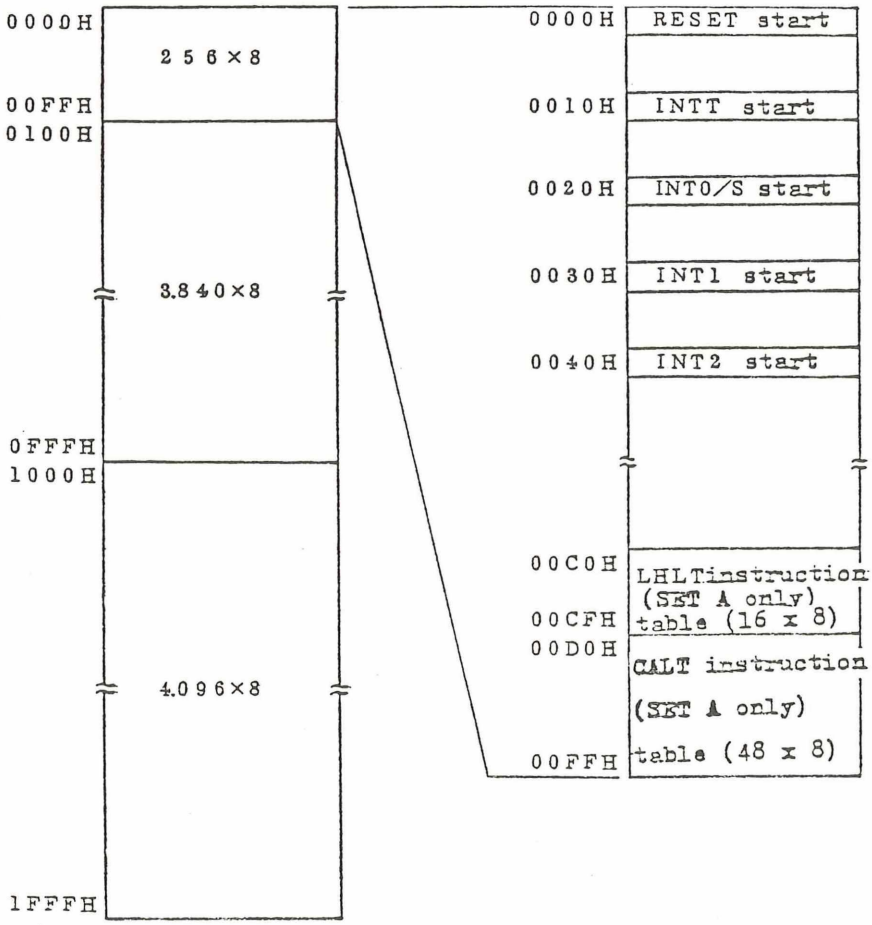
The μPD7500 has two instruction sets, SETA and SETB, and one of which are selected.

4.1 Specific Assignment Address

To the specific locations in 0000H - 00FFH address of bank 0, reset and interrupt start addresses, and the table areas of LHLT instruction (SET A only), CALT instruction (SET A only) are assigned.

Location	Contents of Assignment
0000H	Start of RESET (STOP release/HALT release).
0010H	Start of INTT (STOP release/HALT release).
0020H	Start of INT0/S (STOP release/HALT release).
0030H	Start of INT1 (HALT release).
0040H	Start of INT2 (HALT release).
00C0H S 00CFH	Reference table for LHLT instruction (SET A only).
00D0H S 00FFH	Reference table (call address table) for CALT instruction (SET A only).

Fig. 4-1 Program Memory Map

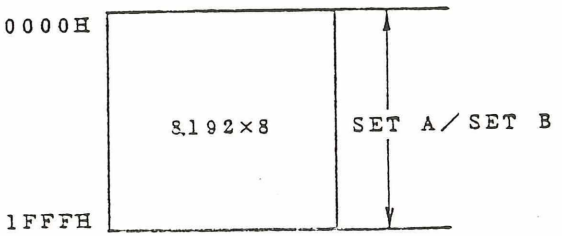


4.2 Jump area of Jump Instructions and Call Instructions

The progam memory area which permits a jump by jump instructions and a call instructions are shown below. There is a difference in jump area between SET A and SET B instructions. So, care should be taken to this point.

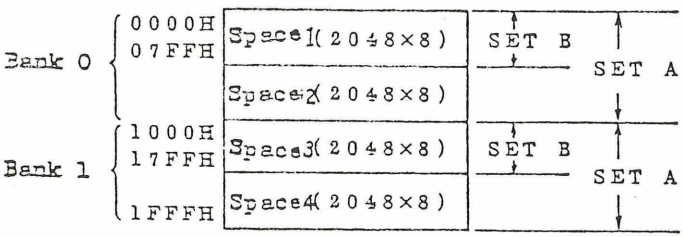
(1) JMPL Instruction

A jump to all the memory area (0000H - 1FFFH) can be made common to SET A/SET B.



(2) JMP Instruction

In the case of SET A, a jump can be made to a desired space (space 1,2 ⟶ space 1, 2, space 3, 4 ⟶ space 3,4) within the bank placed in the JMP instruction. In the case of SET B, a jump is possible to the first half (space 1, 2 ⟶ space 1, space 3, 4 ⟶ space 3) within the bank placed in the JMP instruction.

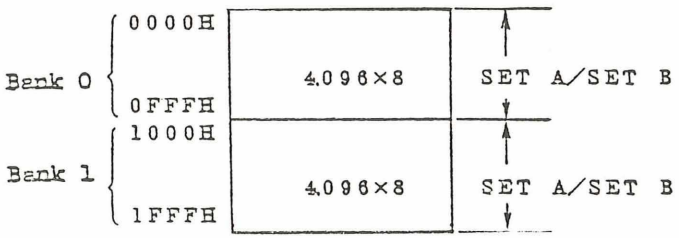


(3) JCP Instruction

A jump can be made to the current page (64 bytes) within the bank placed in the JCP, common to SET A/ SET B. The page is determined by the higher 6 bits of the program counter (PC11 - PC7) at the time of JCP instruction execution. But if the JCP instruction is in the final address of each page, the current page means the next page (as PC11 - PC7 has been incremented by +1). And if JCP instruction is placed in OFFFH, the jumping area is 0000H-003FH and placed in 1FFF, 1000H - 103FH address.

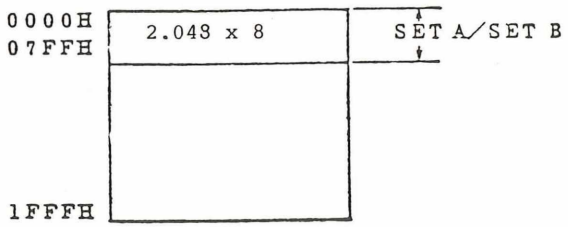
(4) JAM Instruction

A jump is made to all the memory spaces within the bank in which the JAM instruction is located, common to SET A/SET B.



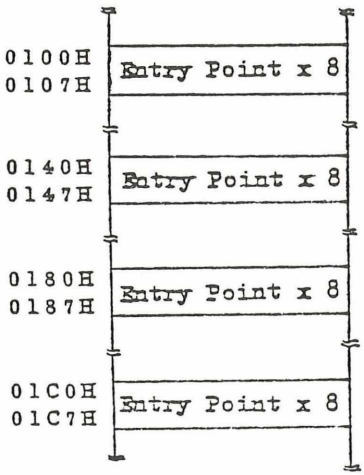
(5) CALL Instruction

This instruction is able to call only the first half (0000H - 07FFH) of bank 0 common to SET A/ SET B.



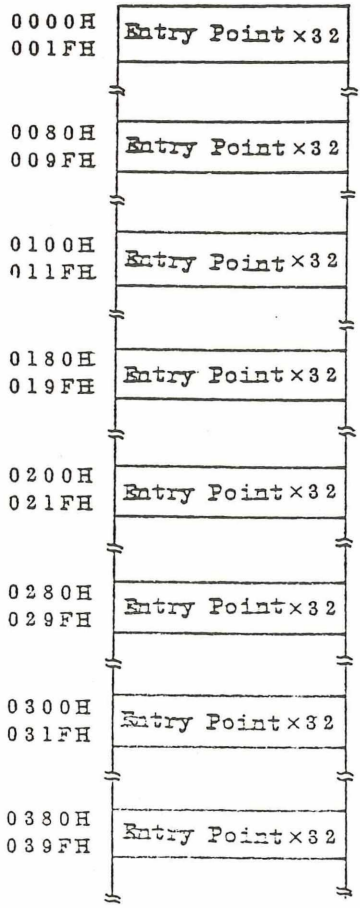
(6) CAL Instruction (SET B Only)

The CAL instruction is specifically provided for SET B, and it can be called only in four blocks (each block is 8 bytes, and interval between blocks is 56 bytes) between 0000H and 01C7H of bank 0.



(5) CALT Instruction (SET A Only)

The CALT instruction is specifically provided for SET A. It can be called only in 7 blocks (each block is 32-bytes length, and an interval between blocks is 96 bytes) between 0000H and 039FH of bank 0.



5.1 Interrupt Service Start Timing

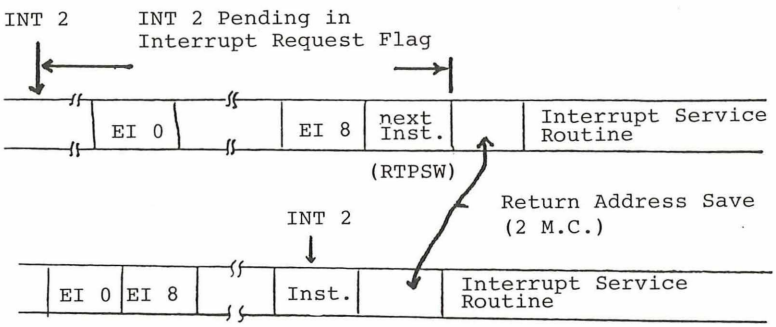
Interrupt service for an interrupt request is started;

i) when the interrupt request is generated while the Interrupt Master Enable F/F and the corresponding bit of the Interrupt Enable Register are both in set state or

ii) when the Interrupt Master Enable F/F and the corresponding bit of the Interrupt Enable Register have both become set while an interrupt request is pending in the Interrupt Request Flag.

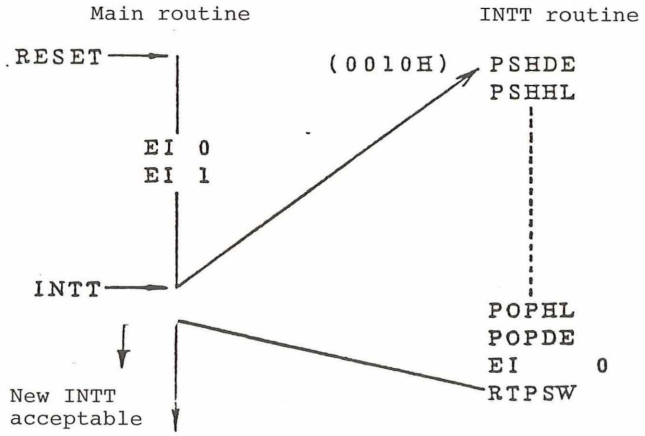
In case of ii), interrupt service operation is started after executing one instruction immediatly following EI 0 or EI n4 (n4≠0) instruction. This operation is convenient to avoid improper increase of stack nesting levels in such case when EI and RTPSW instructions are executed successively at the interrupt service end point and then control is returned to the main routine. Service for a new interrupt request possibly pending in the Interrupt Request Flag is postponed until the RTPSW instruction execution, i.e. until the completion of whole service routine and return of control to the main routine.

Interrupt service start timing is illustrated with the example of INT 2 below. Note that 2 machine cycles are consumed to save the return address into stack just before initiating the interrupt service routine.

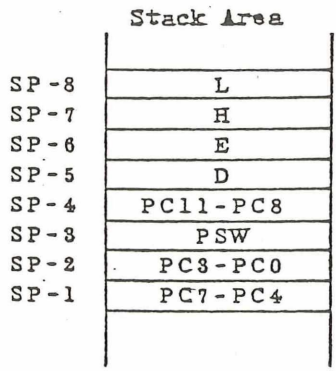


5.2 Example of Interrupt Process Program

An example of INT T is illustrated below.



In this case, the stack is used for interrupt service as below.



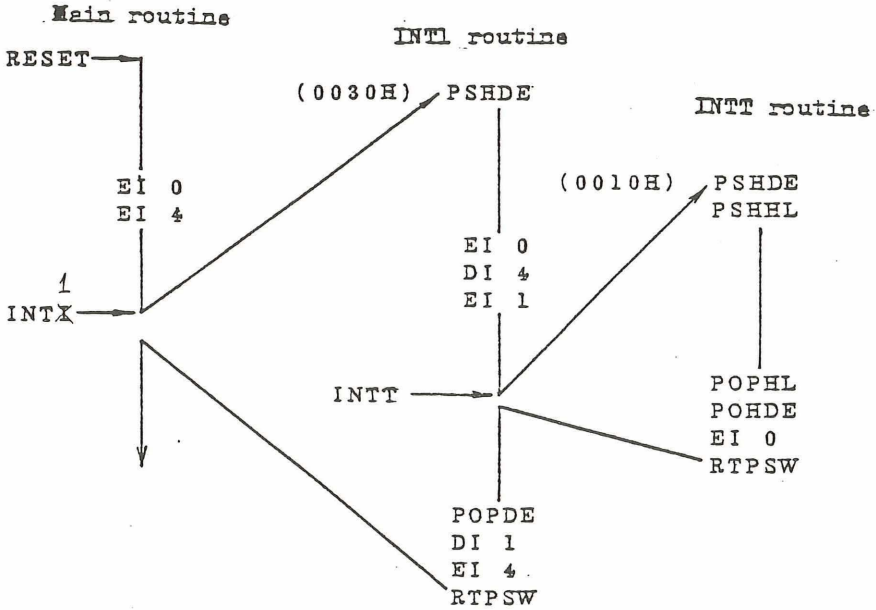
5.3 Multi-level Interrupt

When the interrupt service routine is initiated, the interrupt master enable flip-flop is reset (the same effects as the DI 0 instruction execution), and all interrupts are inhibited. But it can be to accept other interrupt in the interrupt service routine, the EI 0 instruction is executed, and interrupt disable is released. Here, if the corresponding bit of the interrupt enable register is in set state, the interrupt acceptable state is established. The contents of the interrupt enable register are not automatically cleared by an interrupt so that the state before going into the interrupt service routine is effective at that time. Just as in the case of ordinary routines, it is possible to set a desired bit of the interrupt enable register using EI n4 (n4 = 1-F) in the interrupt process routine. Also, unnecessary bits can be reset by using DI n4 (n4 = 1-F) instruction.

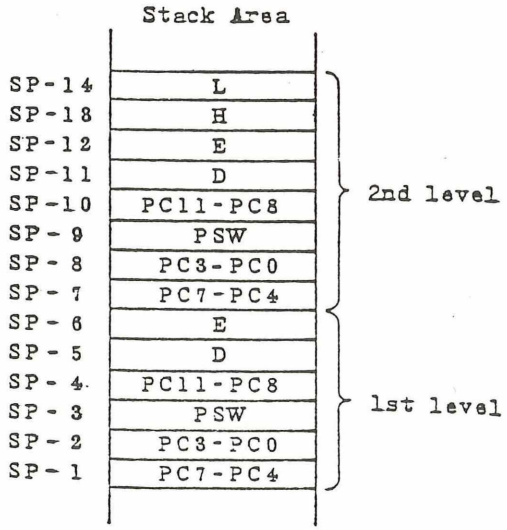
After executing EI 0 instruction in the interrupt service routine, interrupt lower priority than its own can be accepted. Also, when more than two interrupts are generated at the same time, the higher priority interrupt is accepted just as in ordinary cases.

The stack area size depend on the multi-level of interrupt and the number of registers are saved while the interrupt service routine an example of executing the INTT service routine in the INT1 interrupt service routine is shown below.

Fig. 5.1 2 Level Multi Interrupt



In this case the stack is used for interrupt service as below, and 14 words x 4 bits of the data memory are used.



CHAPTER 6. EXPANSION OF INPUT/OUTPUT PORT

The μPD7500 has input and output lines for communication with 8048 family I/O expander (8243), and control instruction, for input/output port expansion purposes.

6.1 Input/Output Lines for 8243 interface

The following input/output lines are used for interface between 8243.

$\overline{\text{CSOUT}}$ output ... 8243 chip select signal.

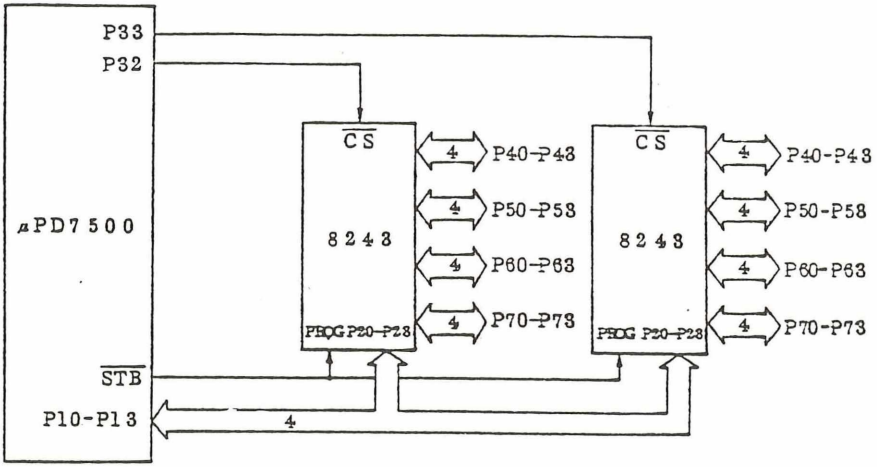
$\overline{\text{STB}}$ or $\overline{\text{PSTB}}$ output ... 8243 PROG signal.

P10 - P13 input/output ... 8243 control command and data transmission (connected to P20 - P23 of 8243)

By using a desired output port of μPD7500 as an 8243 chip select signal output, plural 8243 can be connected.

An example of connecting two 8243 is shown in Fig. 6.1. As a chip select signal output, port 3 (P32,P33) are used.

Fig. 6.1 Example of 8243 Connection



6.2 8243 Control Instructions and Timing

Input instruction : IP, IPL
Output instruction : OP, OPL
Logic output instruction : ANP, ORP

Bit set, reset instructions: SPBL, RPBL

Except for the SPBL and RPBL instructions, these instructions are common to 7500 internal port control instructions, and ports 4 - 7 of 8243 can be selected by designating 8-B.

Also, the bit set, reset instructions designates ports 4 - 7 with higher-order 2 bits of the L register (L3,L2) and makes bit designation with lower-order 2 bits (L1,L0).

Table 6.1 Selection of Ports of Input/Output and Logic Output Instructions

Instruction	Contents of Designation		Port of 8243
IP addr 5 OP addr 3	Operand value	8	Port 4
		9	Port 5
		AH	Port 6
		BH	Port 7
IPL OPL	Contents of L register	8	Port 4
		9	Port 5
		AH	Port 6
		BH	Port 7
ANP addr 6, n4 OPR addr 6, n4	1st operand value	8	Port 4
		9	Port 5
		AH	Port 6
		BH	Port 7

Table 6.2 Selection of Ports and Bits of Set, Reset Instructions by L register

Bit Selection L_1L_0 L_3L_2 Port Selection	00	01	10	11
00	P40	P41	P42	P43
01	P50	P51	P52	P53
10	P60	P61	P62	P63
11	P70	P71	P72	P73

When the 8243 control instruction is executed, a command code for 8243 is output via port 1 of μPD7500 (input to port2 0f 8243), and data transfer between port 2 of μPD7500 and port 2 of 8243. The bit set instruction output the Logical-OR command and specific data (only the designation bit is 1). The bit reset instruction output Logical-AND command and specific data (only designation bit is 0).

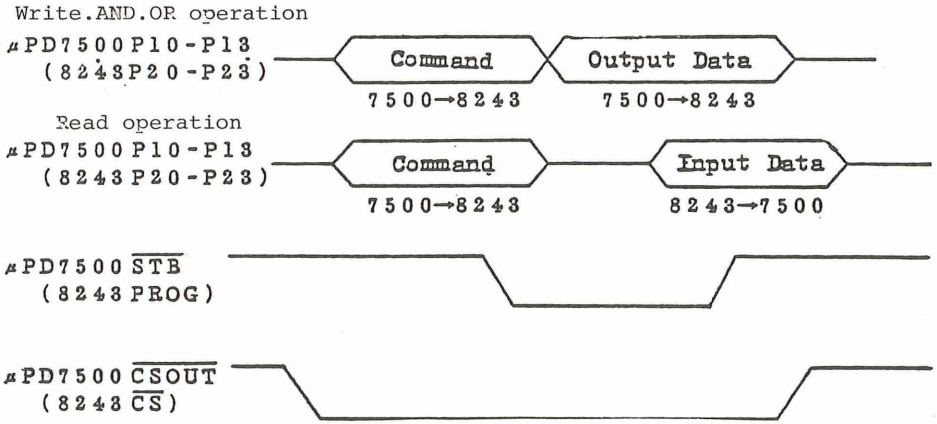
The control of 8243 and port address signal are indicated in Table 6.3 and the input/output timing is shown in Fig. 6.2.

Table 6.3 Control of 8243 and Port Address signal

Port Address P21, P20 / P23,P22 Control Designation	00	01	10	11
00 : Read (port 2 - port n)	Port 4	Port 5	Port 6	Port 7
01 : Write (port n - port 2)				
10 : OR (port n - port 2 $\vee$ port n)				
11 : AND (port n port 2 $\wedge$ port n				

Notes: Port numbers are name of 8243.

Fig. 6.2 Command Output and Data Input/Output Timing



CHAPTER 7. PORT INPUT/OUTPUT TIMING

As instructions related to the port operation of μPD7500, the following types are provided.

Type of Port Operation	Operating Instruction	Timing
Port 1 output	OP, OPL	Section 7.1
Port 2 - 7 output	OP, OPL, ANP, ORP	Section 7.2
Port 3 output only	OP3	
Simultaneous output of ports 4,5	OP54	
Input of ports 0, 1, 4 - 7	IP, IPL	Section 7.5
1 Input only	IPL	
Simultaneous input of ports 4, 5	IP54	
Output of 8243 ports 4 - 7	OP, OPL, ANP, ORP RPBL, SPBL	Section 7.4
Input of 8243 ports 4 - 7	IP, IPL	Section 7.5

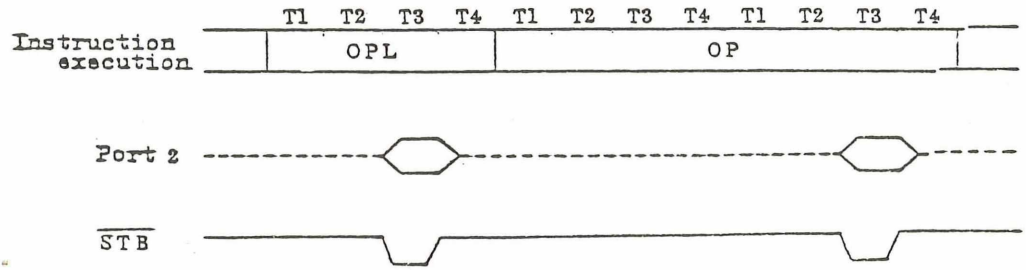
The input and output timings for each operation are shown and described in section 7.1 to 7.5, but for convenience of each instruction explanation, only the execution machine cycle is described. Also, a case of continuously executing 1 byte instruction and 2 byte instruction is also given.

In case of normal port operation, input and output operations are performed in the state of T3 for both the input and output instructions. But in case of 2 byte instruction, input and output operations are made in T3 of the 2nd machine cycle. In case of port 1 output, the strobe signal ($\overline{STB}$) is output in the state of T3.

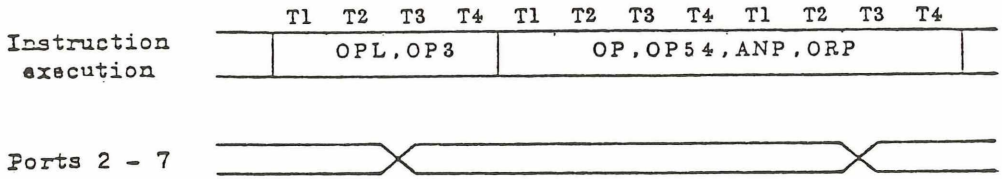
To operate 8243 I/O expander, T2 and T3 states are used, and T2 is the command output timing and T3 is the data input and output timing.

Also, the $\overline{STB}$ signal that becomes the PROG signal of 8243 is output from the end of T2 to T3. The $\overline{CSOUT}$ signal is output as a chip select signal of 8243 continuously between T2 and T3.

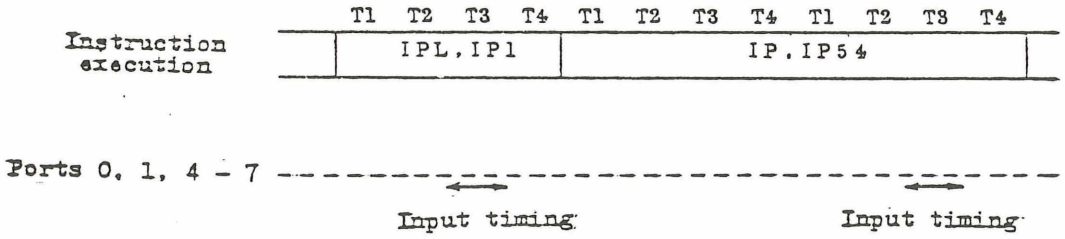
7.1 Execution of OP, OPL Instructions for Port 1



7.2 Execution of OP, OPL, OP3 (Port 3 Only), OP54 (Ports 5, 4 Only), ANP, ORP Instructions for Ports 2 - 7.



7.3 Execution of IP, IPL, IPI (Port 1), IP54 (Ports 5, 4) for Ports 0, 1, 4 - 7



7.4 Execution of OP, OPL, ANP, ORP, RPBL, SPBL for 8243 port 4 - 7

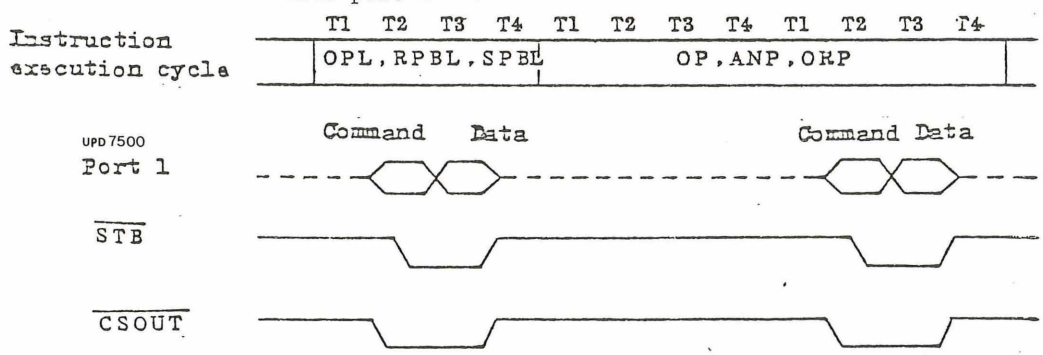
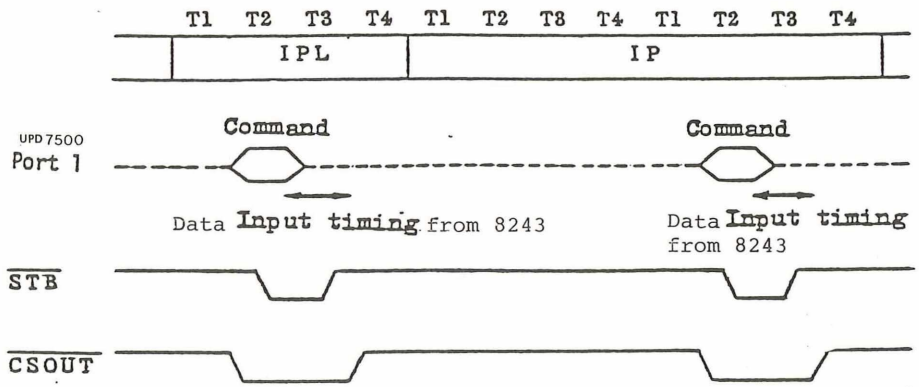


Fig. 7.5 Execution of IP, IPL Instructions for 8243 Ports 4 - 7



CHAPTER 8. INTERFACE FOR BUS OPERATION AND DEBUG

Connection of external program memory and internal information monitor circuit for evaluation is made by using address/data/control buses (BUSO - 13) and three bus control signals (ALE, PSEN, and DOUT).

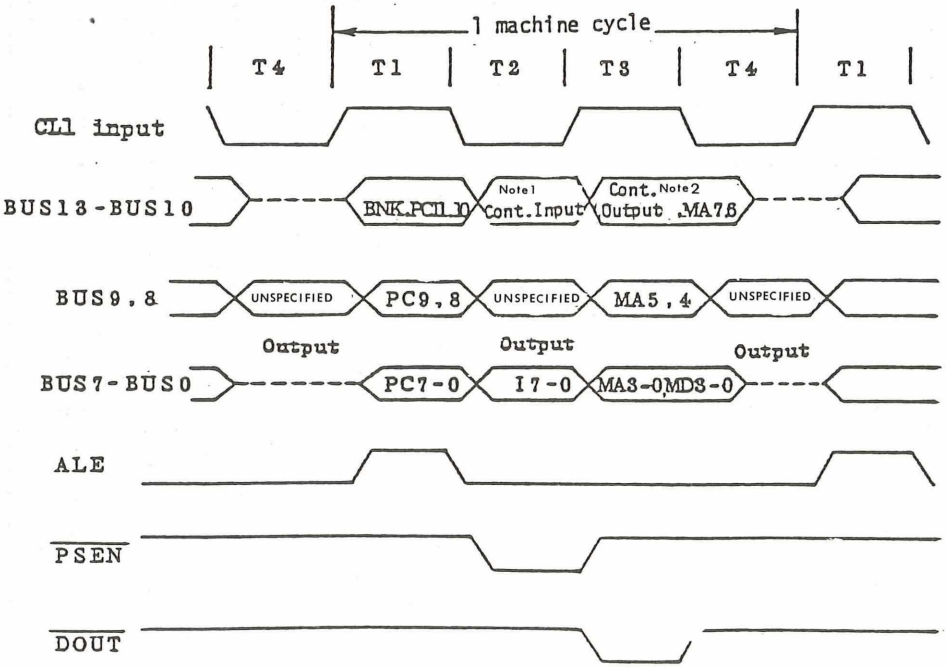
The machine cycle of μPD7500 consists of four different states (T1, T2, T3, and T4). One state is 1/2 clock cycle length, and one machine cycle is a 2 clock cycle length.

8.1 Function of BUS in Each State

BUSO - 13 perform different functions in each of the four states. The operation in T1, T2, T3 are strobed by ALE, $\overline{\text{PSEN}}$, and $\overline{\text{DOUT}}$ signals, respectively. In T4 state, only internal processing is made.

The operation timing of bus is shown in Fig. 8.1, and input/output informations of bus in each state is shown Table 8.1.

Fig. 8.1 Bus Operation Timing



Notes:

1. BRK1,0,DIS I,SET
2. $\overline{SPA}$,$\overline{MW}$

Table 8.1 Bus Input/Output Information of Each State

	T1 (Output)	T2 (Input)	T3 (Output) Operation BRK1=0 BRK0=0	T3 (Output) Break BRK1=0 BRK0=1	T3 (Output) Break BRK1=1 BRK0=0	T3 (Output) Break BRK1=1 BRK0=1
BUS13	*	BRK1	$\overline{SPA}$	1	1	1
BUS12	BNK	BRK0	$\overline{MW}$	1	1	1
BUS11	PC11	DIS I	MA7	H3	D3	SP7
BUS10	PC10	SET	MA6	H2	D2	SP6
BUS 9	PC 9	*	MA5	H1	D1	SP5
BUS 8	PC 8	*	MA4	H0	D0	SP4
BUS 7	PC 7	I7	MA3	L3	E3	SP3
BUS 6	PC 6	I6	MA2	L2	E2	SP2
BUS 5	PC 5	I5	MA1	L1	E1	SP1
BUS 4	PC 4	I4	MA0	L0	E0	SP0
BUS 3	PC 3	I3	MD3	A3	SK1	A3
BUS 2	PC 2	I2	MD2	A2	SK2	A2
BUS 1	PC 1	I1	MD1	A1	A1	A1
BUS 0	PC 0	I0	MD0	A0	C	A0

BNK : Bank flag
PC11 - 0 : Program counter
BRK1, 0 : Break request
DIS I : Interrupt disable request
SET : Instruction set selection
17 - 0 : Instruction code
$\overline{SPA}$: SP addressing
$\overline{MW}$: Data memory write
MA7 - 0 : Data memory address
MD3 - 0 : Data memory write data

H3 - 0 : H register
L3 - 0 : L register
A3 - 0 : Accumulator
D3 - 0 : D register
E3 - 0 : E register
SK1, 0 : Skip flag
C : Carry flag
SP7 - 0 : Stack pointer
* : Unspecify data output

(1) T1 State

In T1 state, the contents of bank flag (BNK) are output to BUS12, and the contents of program counter (PC11-0) to BUS11-0, at the same time the strobe signal (ALE) is output for latching them.

(2) T2 State

In T2 state, the instruction code (Data of external program memory addressed by T1 state output) is fetched via BUS7-0, and the break request signals (BRK1, 0) are fetched via BUS13, 12, the interrupt disable request signal (DIS 1) via BUS11, and the instruction set select signal (SET) via BUS 10.

An instruction code and various control signals read strobe signal ($\overline{PSEN}$) is output at the same time.

(3) T3 State

The operation in the T3 state differs depending upon whether the machine cycle is normal operation cycle, or break cycle.

o Operation Cycle

At the operation cycle, the reference address of internal data memory is output to BUS11-4 (including the time of stack operation), the write data is output to BUS 3-0, and the identification code $\overline{SPA}$, $\overline{MW}$) to define the kind of data on the BUS 11-0 is output to BUS 13 and BUS12.
These BUS data are latched by the strobe signal ($\overline{DOUT}$) which is output in the same time.

o Break Cycle

At the break cycle, the contents of various register are output to BUS11-0, the kind of data on the BUS11-0 is specified by the BRK1, 0 signals which have caused this break cycle. (When BRK1, 0 = 00, the cycle is normal operation cycle).

When BRK1, 0 = 01, the H register is output to BUS11-8, L register to BUS7-4, and the contents of accumulator are output to BUS3-0.

When BRK1, 0 = 10, the D register is output to BUS11-8, E register to BUS7-4, skip flags (SK1, 0) to BUS3-2, bit 1 (A1) of accumulator to BUS1, and the contents of carry flag (C) to BUS0.

When BRK, 0 = 11, the stack pointer is output to BUS11-4, and the contents of accumulator (A) to BUS3-0.

At the break cycle also, various output information can be latched externally using the strobe signal ($\overline{\text{DOUT}}$) output at the same time.

Reference may be made to Chapter 9 for further details of the break operations.

(4) T4 State

The state of T4 is intended for internal processing, and the bus signal and bus control signal become inactive, unspecify data is output to BUS9, 8 and other buses (BUS13-10, BUS7-0) become in a high impedance state.

8.2 Control Signal Input and Output on Bus

(1) BRK1, 0 (Break Request) ... Input

This signal inputs at the T2 state (synchronized with $\overline{\text{PSEN}}$), the signal specify the break operation. When BRK1, 0 = 00, operation is normal. For details of break operation, refer to Chapter 9.

(2) DIS I (Disable Interrupt Request) ... Input

When this signal (High Level active) inputs at the T2 state (synchronized with $\overline{\text{PSEN}}$), the interrupt operation is inhibited. In other words, the interrupt process is not initiated even if an interrupt signal is generated. But since the interrupt request flag is set, it can be tested with the SKI instruction. Then, when the signal becomes low level, the inhibit operation is released.

(3) SET (Select Instruction SET) ... Input

This is a signal for selecting either instruction SET A or SET B of μPD7500. When this signal is made '0'(Low Level)in the T2 state (synchronized with $\overline{\text{PSEN}}$), SET A is selected, and when the signal is made '1' (High Level), SET B is selected.

The input signals of BRK1, 0, DIS I, and SET are not internally latched by μPD7500. So, when it is desired to continue indication, the same data has to be input in the T2 state even after the next machine cycle.

(4) $\overline{\text{SPA}}$ (SP Address), $\overline{\text{MW}}$ (Memory Write) ...

This is an identification signal for defineing the kind of data on the BUS11-0. It is output in the T3 state of the normal operation machine cycle. It can also be latched to the external circuit by $\overline{\text{DOUT}}$ signal.

Table 8.2 Monitoring of Internal Data Memory Operation

$\overline{SPA}$	$\overline{MW}$	Monitoring of Internal Data Memory Operation
1	1	At this time, data to be output to BUS11-0 is undefined.
1	0	When write operation of data in the memory is made indirectly with a register (DE, DL, or HL), or by direct addressing, data memory address (DE, DL, HL, or direct address) is output to BUS11-4 (MA7-0), and write data is output to BUS3-0 (MD3-0).
0	1	When the contents of data memory (stack)are read out by SP indirect addressing, (return, pop instruction), the contents of SP are output to BUS11-4 (MA7-0). The data output to BUS3-0 (MD3-0) is undefined.
0	0	When data is written (subroutine call, push instruction, interrupt) to data memory (stack) by SP indirect addressing, the contents of SP are output to BUS11-4 (MA7-0). The data to be output to BUS3-0 (MD3-0) is undefined.

By checking the contents of SP to be output when $\overline{SPA}$ =0, it is possible to know the data memory area is used for stack, and the stack boundary is strictly observed.

CHAPTER 9. BREAK OPERATION

When the break request signal is input in the T2 of a machine cycle making normal operation, an instruction fetched at the same time is executed immediatly after 1 machine cycle to 3 machine cycle. If the break request signal is continuously being input at each of these machine cycles, the break cycle starts at the end of execution. Just as in the case of normal machine cycles, the break cycle operation consists of four states, namely, T1, T2, T3, and T4. With the exception of T3, the operation is exactly the same as the normal machine cycle.

In the T3, three types of information are output to BUS13-0, depending upon the specify of the break request signal that has caused this break. Once the break cycle starts, it is possible to repeat the break cycles as many times as required if the break request is continued (while varying the specify of BRK1, 0) even during that cycle.

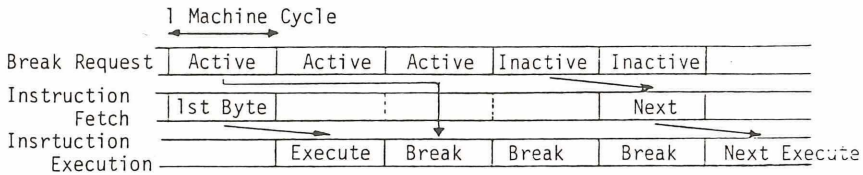
By latching bus output information (see Table 8.1) to an external circuit using the $\overline{\text{DOUT}}$ signal of each machine cycle in the T3 state, the contents of various internal registers can be varified.

In the stand-by mode (STOP/HALT mode), the break request cannot be accepted.

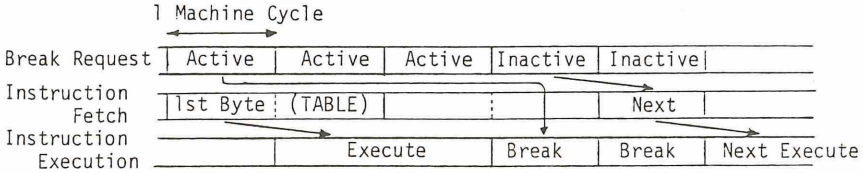
9.1 Break Cycle Timing

The break cycle start timings are different corresponding kind of instructions which fetched at the same time as the break request signal.

(1) Case of 1 Byte 1 Machine Cycle Instruction

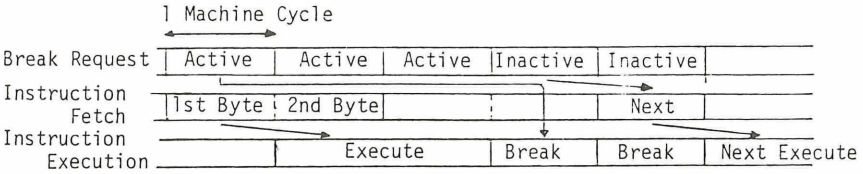


(2) Case of 1 Byte 2 Machine Cycle Instruction

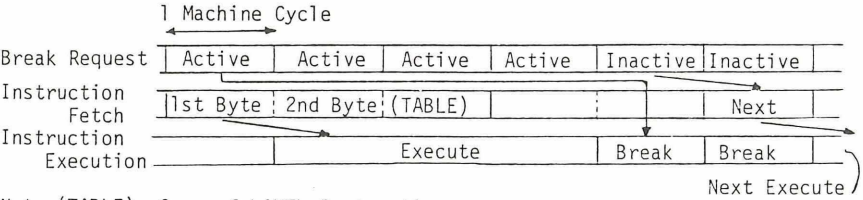


Note (TABLE): Case of CALT,LHLT,LAMT Instruction

(3) Case of 2 Byte 2 Machine Cycle Instruction

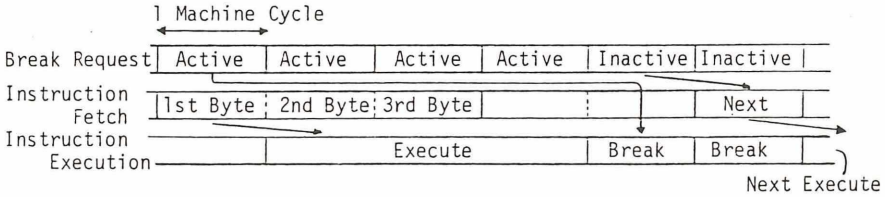


(4) Case of 2 Byte 3 Machine Cycle Instruction

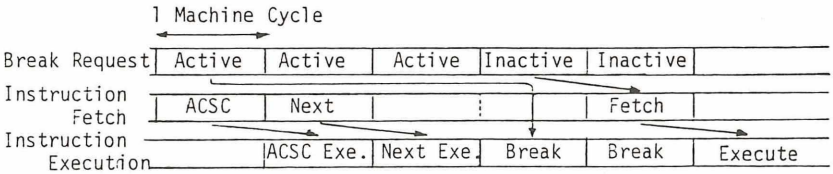


Note (TABLE): Case of LAMTL Instruction

(5) Case of 3 Byte 3 Machine Cycle Instruction

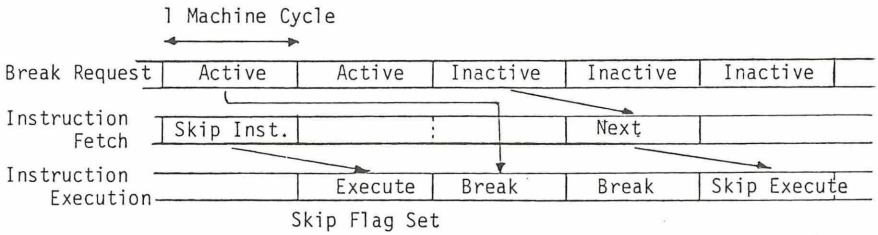


(6) Case of ACSC Instruction



If the instruction fetched at same time as the break request signal is the ACSC instruction, the next one instruction is executed, instead of going into the break cycle immediately after execution of the ACSC instruction.

(7) When Skip Condition of Fetched Instruction is met



If the skip condition of instruction fatched at the sametime as the break request signal is met, the skip flags are set at the end of execution of the instruction, then immediatly the break cycle starts.
Since during operation of break cycle, the skip flags are held, after release of break, the next instruction is skiped in 1 machine cycle.

Since the timing for break differs as above, when the longest time is taken as a common timing (the case of 3 machine cycle instruction), the break request is continuously input during 4 machine cycles including the machine cycle in which the break request is input for the first time. When this is done, a break cycle can be obtained most certainly for all combinations. Thereafter, one machine cycle can be repeated to make it a break cycle.

9.2 Registers Held During a Break Cycle

The break cycle is initiated, the contents of various registers listed below are held until the break request has been released.

D, E, H, L registers
Accumulator (A)
Stack Pointer (SP)
Skip flags (SK1, 0)
Carry flag (c)

The contents of these register flag can be read out by three break cycles (BRK 1, 0 = 10, 01, 11), but so long as the break cycle continues, the same result is obtained in whatever sequence read operation is made.

9.3 Interrupt During Break Cycle

When an interrupt has occurred during the break cycle, if the interrupt control circuit is in the interrupt enable state and also if DIS I = 0, a jump is made to the interrupt start address, and the break cycle continues without executing the instruction. When the break request is released, an interrupt service program is executed from the interrupt start address. When DIS I = 1 is independent the interrupt control circuit status and the break cycle continues with the interrupt request flag set.

9.4 RESET During Break Cycle

When RESET has occured during the break cycle, a jump is made to the RESET start address (0000H), and the break cycle continues with-out executing the instruction, after release of break, the RESET program is executed from the RESET start address.

CHAPTER 10. CPU CONTROL FUNCTION

10.1 Stand-by Function

Two types of stand-by mode (STOP mode and HALT mode) are provided for μPD7500 so as to save the power consumption during a program stand-by state. The STOP mode is set, by the STOP instruction and the HALT mode by the HALT instruction. When the stand-by mode is set, program execution is stopped, and the contents of all internal registers and the contents of data memory are held unchanged.

In the stand-by mode, it is possible to operate the shift register and the timer/event counter.

To release the stand-by mode, the interrupt generation or RESET input is available.

Because an interrupt is used for release of the stand-by mode, neither the STOP mode nor the HALT mode can be set though the STOP instruction or HALT instruction is executed if even one interrupt request flag is set. Therefore, when setting the stand-by mode at a point where the possibility of a request flag being set, it is necessary to have the interrupt request flag reset by processing the interrupt in advance or by executing the SKI instruction.

Comparison between the STOP mode and the HALT mode is shown in Table 10.1, but a biggest difference lies in that either RC oscillation output (CL) stops (in the STOP mode) or not stop (in the HALT mode) when a clock is being supplied by RC oscillation.
In stop mode, it is possible to retain the contents of the data memory if power supply becomes low.

Table 10.1 Comparison between STOP Mode and HALT Mode

	Set Instruction	CL	ø	X	CPU	SIO	CNT	Interrupt used for release.
STOP Mode	STOP	×	×	○	×	△	△	INTT, INT0/S
HALT Mode	HALT	○	×	○	×	△	○	INTT, INT0/S, INT1, INT2

○ - Operation possible.

△ - Operation possible with a mode selected.

X - Stop

10.1.1 STOP Mode

In the STOP mode, the system clock generator circuit stops (RC oscillation and 1/2 frequency dividing operation stop). So, the operation requiring the CPU and the system clocks (CL,ø) comes to a stop. But the count clock generator circuit and the clock control circuit are in operation even at that time, the timer/event counter can be operated if count clock (X) is selected for count pulse (CP). By the same selection, the LCD clock signal continues to be output from the LCD CL pin. Also, the serial interface can be normally operated if a serial clock is selected to other than ø (external clock, TOUT signal is available).

The release of STOP mode can be achieved by RESET input, and generation of INTT, INTO/S interrupt (irrespectively of whether the state is interrupt disable or enable).

Incidentally, when system clock generation is driven by external CL1 input, CL does not stop, and ϕ only stops. So, the result is the same state as the HALT mode described next. Release is made much in the same manner as in the HALT mode. Therefore, execution of the STOP instruction for setting the STOP mode comes to carry a meaning at the time of RC oscillation.

10.1.2 HALT Mode

In the HALT mode, 1/2 divider circuit in the clock generator circuit only stops (CL operates, ϕ stops). So, the operation of the CPU and serial interface (when using ϕ as a serial clock) requiring ϕ signal is halted.

But as the count clock generator circuit now operates, and as the clock control circuit is in operation also, the clock control circuit which inputs are the CL signal from the clock generator circuit and the X signal from the count clock generator, selects these inputs and supplies the count pulse (CP) to the timer/event counter and the LCD clock signal to LCD CL pin. As a result, the timer/event counter can be operated, so that the timer operation never stops.

The serial interface operates normally if other than ϕ (external clock, TOUT signal) is selected as a serial clock. The HALT mode can be released by generation of RESET input and generation of all interrupts (INTT, INTO, INTS, INT1, INT2) irrespectively of the interrupt disable and enable.

10.1.3 The Release of Stand-by mode by an Interrupt

The release of the stand-by mode due to an interrupt is made when the interrupt request flag is set by each interrupt source. At that time, the release is occured whether interrupt is disable or enable state. So, release can be made, but operation after release differs from each other.

(1) When Interrupt Master Enable F/F = 1

If the corresponding bit of the interrupt enable register, corresponding to the release interrupt has been set, the corresponding interrupt routine is initiated after execution of one instruction after STOP/HALT instruction and the request flag is reset then.

If the corresponding bit of the interrupt enable register has been set, the execution of an instruction after the STOP/HALT instruction is initiated (the interrupt routine is not made). At that time, the request flag used for release remains set. So, reset of request flag has to be made by the SKI instruction, is necessary.

(2) When Interrupt Master Enable F/F = 0

An instruction following the STOP/HALT instruction begins to be executed irrespectively of the interrupt enable register. (Interrupt routine is not initiated.) At that time, the request flag used for release has been left set. So, it has to be reset by the SKI instruction, if necessary.

In any release function, the operation can be resumed by the contents immediately before the stand-by mode has been selected.

10.1.4 Release of Stand-by Mode by RESET Input

The stand-by mode, namely, the STOP mode and the HALT mode, is released unconditionally by the RESET input. The release timing is shown in Fig. 10.1

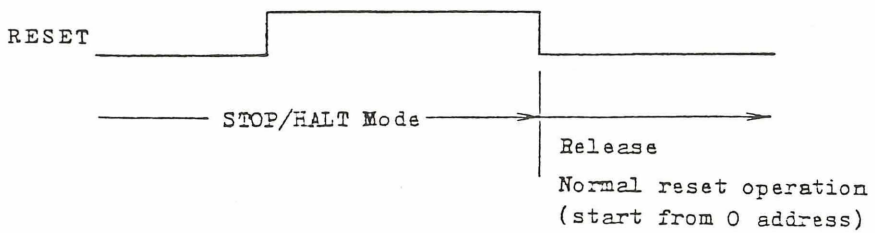


Fig. 10.1 Release Timing for Stand-by Mode by RESET Input

During the active (high level) state of RESET input, the stand-by mode is continued, and when RESET input goes low, the stand-by mode is released. After normal reset operation, the program is started from 0 address.

The contents of data memory remains unaffected by RESET input in the stand-by mode.

10.2 Reset Function

The μPD7500 is reset and initialized as below by input of a high level active RESET signal.

10.2.1 Contents of Initialization of CPU

(1) The program counter (PC11-PC0) is cleared to 0.

(2) The bank flag (BNK) and skip flag (SK1, SK0) in program status words are reset (0).

(3) The timer/event counter becomes as below.

- o Count register = OOH
- o Modulo register = FFH
- o Timer out flip-flop = 0

(4) The clock control circuit becomes as below.

- o Clock mode register (CM3 - CM0) = 0

  $CP = LCD\ CL = \frac{CL}{256}$

  the timer out F/F signal is not output to PTOUT pin.

- o Prescalers 1, 2, 3 are cleared to 0.

(5) Shift Mode Registers (SM3 - SM0) are cleared to 0.

Shift operation stops.
Port 0 is in high impedance state.
INTS is selected by INT O/S request selector.

(6) The interrupt control circuit becomes as below.

- o Interrupt request flags are cleared to 0
- o Interrupt master enable F/F = 0
- o Interrupt enable register = 0000B
- o INT1 mode registers (IM1, IM0) = 0

All hold interrupts disappear.
All interrupts disabled.
INT1 input is the rising edge active.

(7) Port 6 mode registers (PM3 - PM0) are cleared to 0.

(8) All output buffers of ports 0 - 7 are turned off, to be placed into a high impedance state. The input/output ports are set to input mode.

(9) The contents of the data memory and those of below-mentioned registers are undefined.

Stack pointer (SP)
Accumulator (A)
Carry flag (C)
General-purpose registers (D, E, H, L)
Output latches of each port
Shift register

When RESET input is released, the program starts from 0000H address, so the contents of these registers should be initialized by the program.

10.2.2 Power-ON-Reset Circuit

As the most simple reset control circuit, an example of the power-on-reset circuit using a resistor and a capacitor is shown in Fig. 10.1.

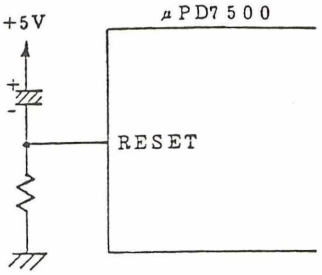


Fig. 10.1 Power ON Reset Circuit

CHAPTER 11. EXAMPLE OF SYSTEM CONFIGURATION

11.1 Connection of Program Memory

An example of connecting one chip of 2716 (EPROM 2,048 words x 8 bits) as a 2 K byte program memory is shown in Fig. 11.1. As an address signal latch, three chips of μPD4042BC (quad clocked "D" latch) are employed and ALE output are used as a latch signal. In the 2716 $\overline{OE}$ (Output Enable) signal, $\overline{PSEN}$ output is used.

11.2 Register Dump Circuit

An example of a circuit for driving the LED lamp using three chips of μPD4175BC (quad D-type flip-flop) and two pieces of μPA2003C (6 element transistor array).

By changing the break request signals (BRK1, 0) to be input to BUS13, 12 via μPD4503BC (Hex. Bus Driver) by means of switch, the contents of various registers can be displayed on the LED lamp in units of bit.

As a clock signal of μPD4175BC, $\overline{DOUT}$ output is used, as a DIS A signal of μPD4503BC, $\overline{PSEN}$ output is used.

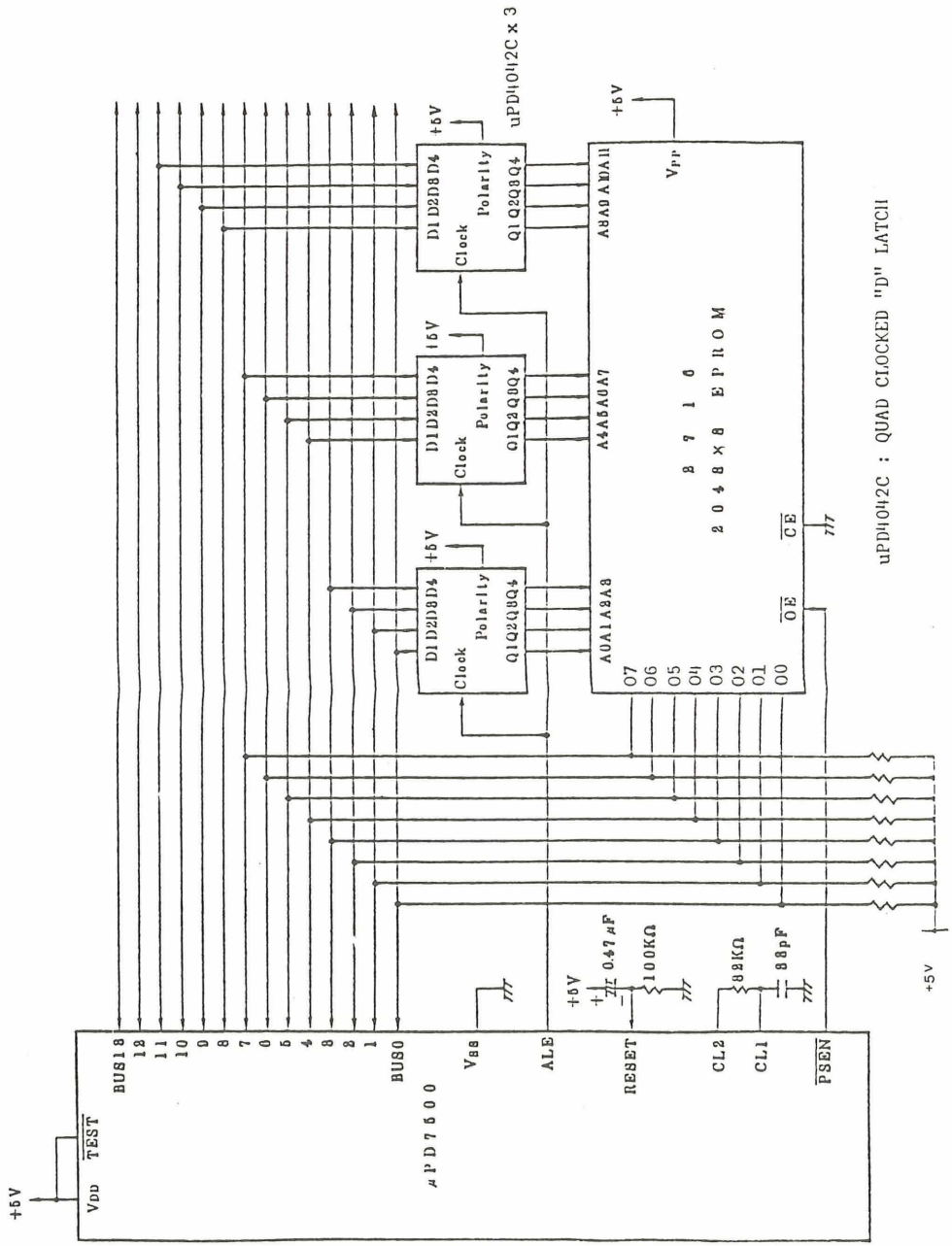
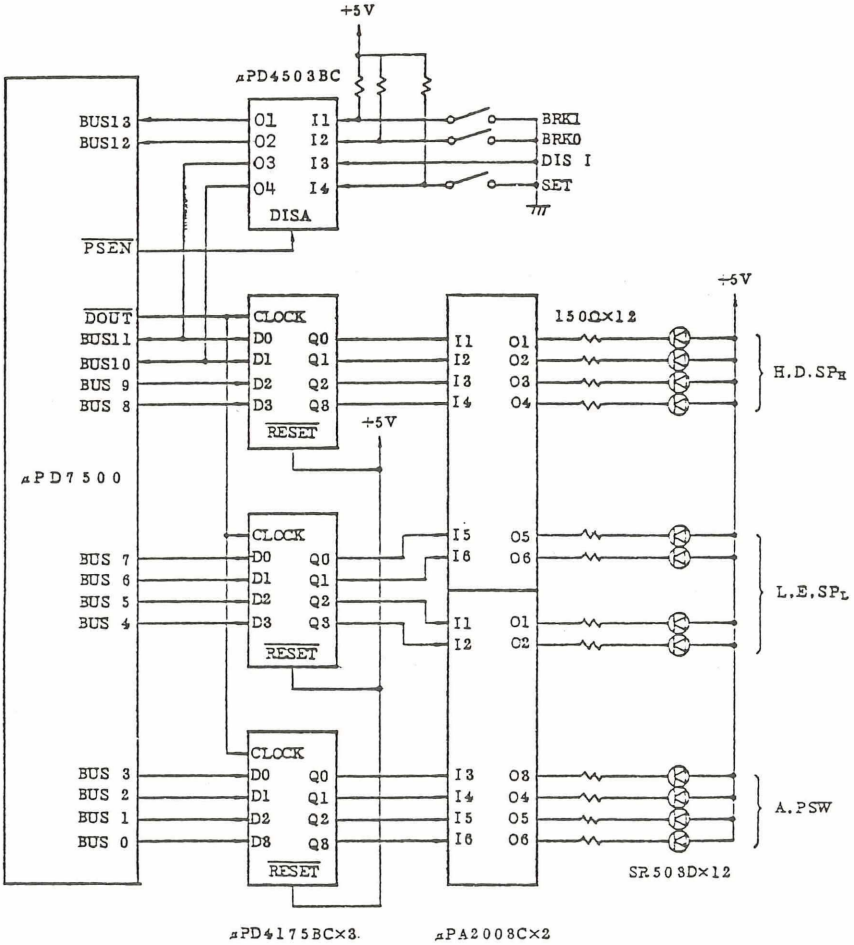


Fig. 11.1 Example of Program Memory (2716) Connection (For References)



μPD4175BC : Quad. D-Type Flip-Flop
μPD4503BC : Hex. Bus Driver
μPA2003C : Transistor Array
SR503D : LED Lamp

Fig. 11.2 Example of Register Dump Circuit Configuration

11.3 Break Control Circuit

Fig. 11-3 illustrates a break control circuit example that enables a break of program execution at any desired address (stop address) and also the single step operation.

First, the stop address is set to the 12 switches (ON = 0, OFF = 1). Program break will occur immediately after completing execution of the instruction that contains the byte at the location specified by the stop address. Address output signals from the μPD7500 BUSO-BUS11 are latched by the μPD4042C's (Quad Clocked D-Latch), and their latch outputs are compared with the stop address set to the 12 switches by the μPD4063C's (4 Bit Magnitude Comparator). When a match is detected, the μPD4063C's will output a match signal (A=B:OUT), which sets the μPD4043C (Quad 3-State NOR R/S Latch). The Q1 output of the μPD4043C is input to I3 of the μPD4503BC (Hex Bus Driver) via an AND gate.

The 03 output of μPD4503BC is ON while DisA input is low ($\overline{\text{PSEN}}$=low), and transfers I3 input signal to the μPD7500 BUS12 as the BRKO signal. By this, the break request signals become BRK1, 0 = 0, 1, and the μPD7500 enters the break cycle.

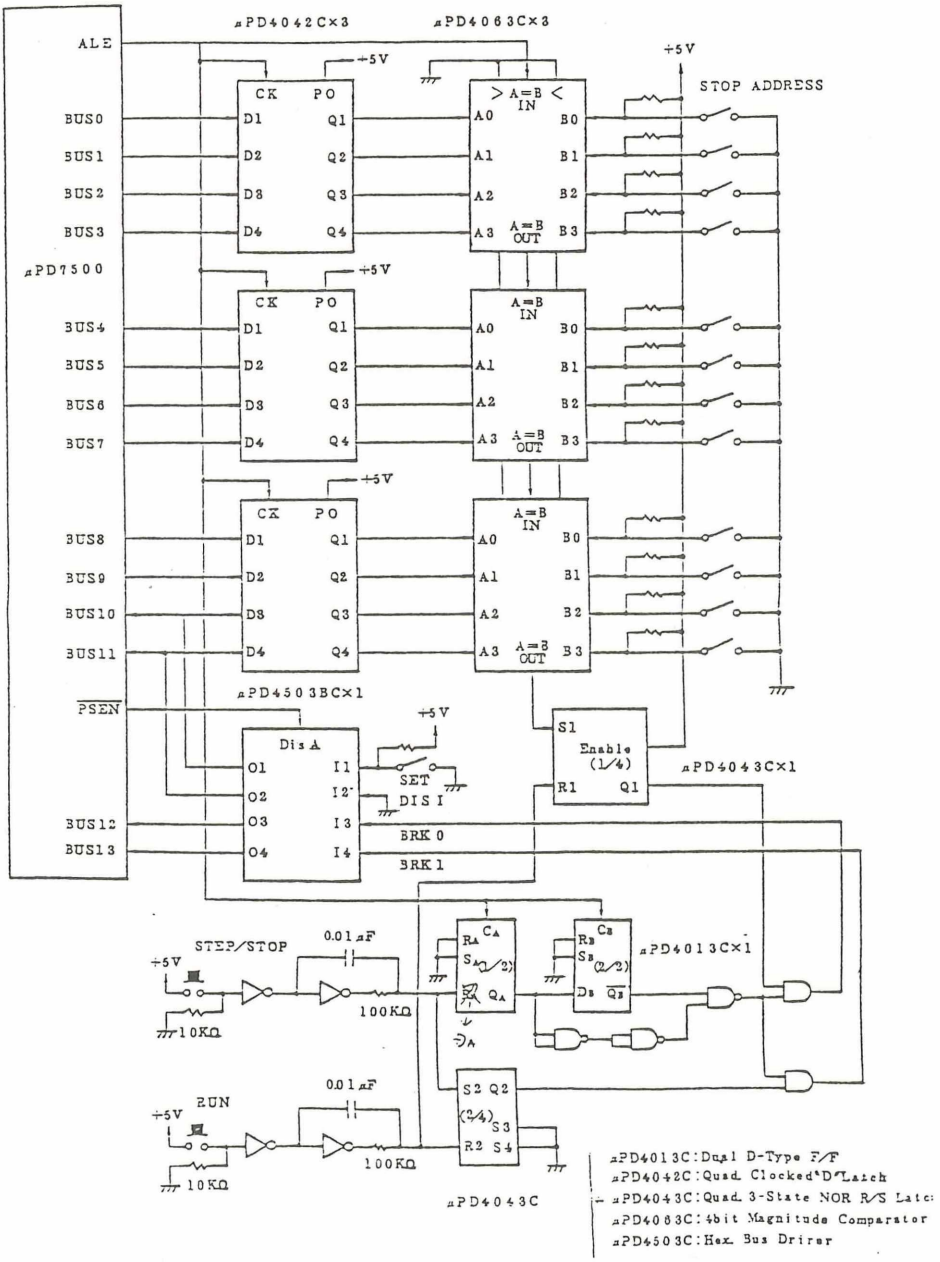
By pressing STEP/STOP key once, while in the break cycle, the break request is released for 1 machine cycleperiod and then break request signals (BRK1, 0 = 1, 1) are reactivated by the break request control logic that uses the μPD4013C (Dual D-Type F/F) and the ALE signal as the clock. With this operation, program execution is advanced by 1 step (1 instruction).

After a break occurs, single step operation can be performed by simple pressing the STEP/STOP key in this manner, single step instruction execution at each key press.

The break cycle is released by pressing the RUN key. When RUN key is pressed, all the break request signals are reset, and the program execution is restarted and continued until next break request occurs.

While in program run mode, break cycle (BRK1, 0=1,0) can also be generated by pressing the STEP/STOP key. After the break cycle is initiated, single step operation can be performed with the STEP/STOP key in the exactly same manner as after the preset break, and the break request signals also become BRK1, 0=1,1.

In Fig. 11-3, a momentary key is used for the STEP/STOP key and RUN key. An inverter and a feedback capacitor are used to trim the waveform and eliminate chattering.


μPD4042C×3
μPD4063C×3
ALE
+5V
STOP ADDRESS
CK
PO
D1
Q1
D2
Q2
D3
Q3
D4
Q4
A0
B0
A1
B1
A2
B2
A3
B3
A=B IN
A=B OUT
BUS0
BUS1
BUS2
BUS3
μPD7500
BUS4
BUS5
BUS6
BUS7
BUS8
BUS9
BUS10
BUS11
μPD4503BC×1
PSEN
Dis A
O1
O2
O3
O4
I1
I2
I3
I4
SET
DIS I
BRK 0
BRK 1
BUS12
BUS13
S1
Enable (1/4)
R1
Q1
μPD4043C×1
STEP/STOP
0.01μF
10KΩ
100KΩ
RUN
μPD4013C×1
S2 Q2
S3
R2 S4
μPD4043C
μPD4013C:Dual D-Type F/F
μPD4042C:Quad Clocked D Latch
μPD4043C:Quad 3-State NOR R/S Latch
μPD4063C:4bit Magnitude Comparator
μPD4503C:Hex Bus Driver

11.4 Expander Circuit of Data Memory

An example of a circuit for externally expanding the data memory using port 1 (strobed 4 bit input/output port) is shown in Fig. 11.4. As data memory, μPD444 (1,024 words x 4 bit static CMOS RAM) is used.

As address input (AO - A9) of μPD444, the higher-order 2 bits of port 2 (P22, P23), port 3 (P30 - P33), and port 4 (P40 - P43) outputs are used, and as data input/output (I/O_1 - I/O_4), port 1 (P10 - P13) are used.

The μPD444 outputs data onto the I/O 1-4 merely with $\overline{CS}$ brougt low, and when $\overline{WR}$ input is forced low at

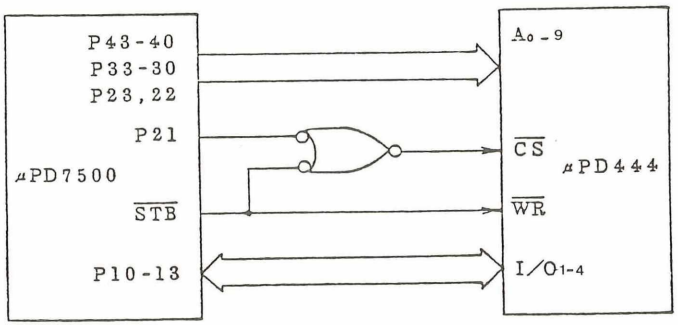


Fig. 11.4 Example of Interface with uPD444 (For References)

the same time as $\overline{CS}$ becoming low, the data on the I/O 1-4 line is written (in write operation). To make read operation with this connection, the low-level read strobe signal (chip select signal) is output from P21 output after output of address signal from ports 4,3, and 2, thereby executing the port 1 input instruction.

To make write operation, the address signal is output from ports 4, 3, and 2, to execute the port 1 output instruction after output of a high level signal to P21. By the port 1 output instruction, the write strobe signal and write data are output to $\overline{STB}$ pin and P10 - P13 pins, respectively.

11.5 Interface with μPD549

The μPD549 is an 8 digit, 8 segment output type programmable display controller for changing graphics pattern by software.

The μPD549 controls the command/data input from S_0-S_3 terminals by the control signal (CTL). The command/data can be written by CTL signal, that is, without using the write control signal ($\overline{WR}$), but when controlling it by the μPD7500, the mode using the $\overline{WR}$ signal is available with use of port which can deliver strobed output.

The CTL signal uses port 2. In this example, two chips of μPD549 are used, the higher-order 2 bits (P22, P23) of the port 2 are assigned to the respective CTL signals.

Incidentally, as the μPD549 is a PMOS device (operating on - 10V power supply), to make actual connection, it is necessary to consider level interface. An example of μPD7500 and two chips of μPD549 connected is shown in Fig. 11.4.

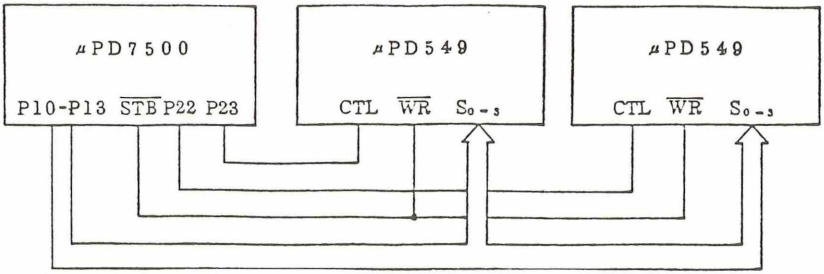


Fig. 11.5 Example of Interface with μPD549
(For References)

12.1 Operand Expression Form and Method of Description

The operand is described in the operand field of each instruction according to the method of description corresponded with the operand notations. (For further details, refer to Assembler Specification). In the method of description, one of the elements should be selected if there are a plurality of such statements. Alphabet written in Capital letter and +, - symbols are key words, which should be written as are.

In the case of immediate data, proper value or label has to be indicated.

Expression	Method of Description
addr	13 bit immediate data or label
addr1	bit immediate data or label
addr2	bit immediate data or label
addr8	4 bit immediate data or label
addr4	1 - FH immediate data or label
addr5	0,1,4-BH immediate data or label
addr6	2 - BH immediate data or label
caddr	11 bit immediate data or label
caddr1	0100H-0107H,0140H-0147H, 0180H-0187H,01C0H-01C7H
taddr	00C0H-00CFH
taddr1	00D0H-00FFH
mem	8 bit immediate data or label
byte	8 bit immediate data or label
n5	5 bit immediate data or label
n4	4 bit immediate data or label
n3	3 bit immediate data or label
bit	2 bit immediate data or label
pr	DL,DE,HL-,HL+,HL
pr1	HL-,HL+,HL

12.2 Legend Used for Operation Description

A	:	Accumulator
D	:	D register
E	:	E register
H	:	H register
L	:	L register
DE	:	Pair register (DE)
DL	:	Pair register (DL)
HL	:	Pair register (HL)
pr	:	Pair register (DL, DE, HL-, HL+, HL)
sp	:	Stack pointer
PC	:	Program counter
CT	:	Count register
BNK	:	Bank flag
C	:	Carry flag
PSW	:	Program status word
SIO	:	Shift register
MOD	:	Moduro register
IE	:	Interrupt enable register
IME	:	Interrupt master enable F/F
In	:	Immediate data for byte, n5, n4, n3
Dn	:	Immediate data for addr 4, addr 5, addr 6, mem.
Bn	:	Immediate data for bit
Rn	:	Immediate data for pr, prl
(xx)	:	Contents addressed by xx.
xxH	:	Hexadecimal data
Pn	:	Immediate data for addr, addr 1, addr 2, addr 3, caddr, taddr, taddr 1.

12.3 SET A and SET B

For the μPD7500, two types of instruction sets (SET A and SET B) are provided. The user is therefore able to use one of them by selection.

And SET A group is larger than SET B group in volume.

In the description of instructions, these instructions are distinguished by indicating SET A or SET B to the right side of the instruction mnemonic.

SET A	: SET A instructions, but in SET B there are same mnemonic instructions which function different from the instructions of SET A mnemonic.
SET B	: SET B instructions, but in SETA there are same mnemonic instructions which function different from the instruction of SET B.
SET A only	: Instructions which are contained SET A group only.
SET B only	: Instructions which are contained SET B group only.
Nothing indicated	: Instructions available in both SET A and SET B.

12.4 Bank Boundary

Bank 0 and bank 1 are designated by the contents of bank flag (BNK) independent of the contents of program counter. Therefore, it is not possible to place 2 byte or 3 byte instruction on bank boundary (the bank 0 and bank 1).

12.5 Instruction Set

12.5.1 Load/Store Instruction

LA1 n4 (Load A with Immediate)

1 Instruction code = $\boxed{0\ 0\ 0\ I\ I_3\ I_2\ I_2\ I_o}$

2 Bytes = 1

3 Cycles = 1

4 Function = A - n4 n4 = I_{3-0} = 0 - $F_{(16)}$

This instruction loads 4 bits of immediate data into the accumulator. The LAI instruction has a string effect. If the same instructions appear in sequence as a string, only the first will be executed. The 2nd instruction and the remainder in the instruction string are executed as NOP (No Operation) instructions.

5 Example: This is an example of subroutine for realizing that the contents of Z address add to one or two or three.

```
ZAD1 = LAI     1 = (1)
ZAD2 = LAI     2 = (2)
ZAD3 = LAI     3 = (3)
       LHLI  Z
       ASC
       NOP
       XAM HL
       RT
```

(1) When ZAD1 is Called

1 is loaded into the accumulator with the LAI 1 instruction, and the second (2) and third (3) are executed as NOP. Thus, 1 is binary added to the contents of the data memory in the address location Z.

(2) When ZAD2 is Called

2 is loaded into the accumulator with the LAI 2 instruction, and the third (3) instruction is executed as NOP. Thus, 2 is binary added to the contents of the data memory in the address location Z.

(3) When ZAD3 is Called

3 is loaded into the accumulator with the LAI 3 instruction, and 3 is binary added to the contents of the data memory in the address location Z.

LD1 n4 (Load D with Immediate) .... SET A only

1 Instruction code =

2 Bytes = 2

3 Cycles = 2

4 Function = D - n4 n4 - I_{3-0} = 0 - $F_{(16)}$

This instruction loads 4 bits of immediate data n4 into the D register.

LEI n4 (Load E with Immediate) .... SET A only

1 Instruction code = | 0 0 1 1 1 1 1 0 | | 0 0 0 0 I_3 I_2 I_2I_0 |

2 Bytes = 2

3 Cycles = 2

4 Function = E - n4 n4 = I_{3-0} = 0 - $F_{(16)}$

This instructions loads 4 bits of immediate data into the E register.

LHI n4 (Load H with Immediate) .... SET A

1 Instruction code = | 0 0 1 1 1 1 1 0 | | 0 0 1 1 $1_3I_2I_1I_0$ |

2 Bytes = 2

3 Cycles = 2

4 Function = H - n4 n4 = I_{3-0} = 0 - $F_{(16)}$

This instruction loads 4 bits of immediate data into the H register.

LHI n3 (Load H with Immediate) .... SET B

1 Instruction code = | 0 0 1 0 1 1_2 I_1 I_0 |

2 Bytes = 1

3 Cycles = 1

4 Function = H_{2-0} - n3, H_3 - 0 n3 = I_{2-0} = 0 - 7

The LHI n3 instruction loads 3 bit immediate data n3 to low-order 3 bits (H_{2-0}) of the H register. The most significant bit (H_3) of the H register is unconditionally loaded with zero.

LLI n4 (Load with Immediate) .... SET A only

1 Instruction code = | 0 0 1 1 1 1 1 0 | 0 0 0 I I_3 I_2 I_1 I_0 |

2 Bytes = 2

3 Cycles = 2

4 Function = L n4 - n4 = I_{3-0} : 0 - $F_{(16)}$

This instruction loads 4 bits of immediate data into the L register.

LAM pr (Load A with Memory) .... SET A

1 Instruction code = | 0 I 0 R_+ 0 0 R_1 R_0 |

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = A - (pr)
When pr = HL+ = skip if L = 0
When pr = HL- = skip if L = $F_{(16)}$

R_+ R_1 R_0	pr
0 0 0	DL
0 0 1	DE
1 0 0	HL-
1 0 1	HL+
1 1 0	HL

This instruction loads the contents of the memory addressed by a pair register into the accumulator. When HL+ is specified with pr, the contents of the L register are incremented after loading the data. So, when the result is 0, the succeeding one instruction is skipped. If HL- is specified with pr, the contents of the L register are decremented after loading the data. So, when the result is $F_{(16)}$, the succeeding one instruction is skipped.

LAM prl (Load A with Memory) .... SET B

1 Instruction code = | 0 1 0 1 0 0 R_1 R_0 |

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = A - (prl) prl = HL-, HL+, HL

When rpl = HL+, skip if L = 0

When prl = HL-, skip if L = FH

prl	R_1 R_0
HL−	0 0
HL+	0 1
HL	1 0

This instruction loads the contents of data memory addressed by pair registers (HL-, HL+, HL) to the accumulator. When HL+ is designated by prl, the contents of L register are incremented (+1) after data loading. If the result is 0, the succeeding one instruction is skipped. When HL- is designated by pr 1, the contents of L register are decremented (-1) after data loading. When the result is FH (15), the succeeding one instruction is skipped.

6 Note : LAM HL-, LAM HL+, LAM HL can be described also as LDS, LIS, L (without operand in all these cases), respectively.

LADR mem (Load A Direct)

1 Instruction code = | 0 0 1 1 1 0 0 0 | $D_7D_6D_5D_4D_3D_2D_1D_0$

2 Bytes = 2

3 Cycles = 2

4 Function = A (mem) mem = D_{7-0} = 00 - $FF_{(16)}$

This instruction loads the contents of the data memory addressed with 8 bits of immediate data (mem) into the accumulator.

5 Example:

The contents of data in the memory location $6F_{(16)}$ is loaded into the accumulator.

LADR 6FH = A (6F)

LDEI byte (Load DE with Immediate) .... SET A only

1 Instruction code = 0 1 0 0 1 1 1 1 | $I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0$

2 Bytes = 2

3 Cycles = 2

4 Function = DE - byte byte = I_{7-0} = 00 - $FF_{(16)}$

This instruction loads 8 bits of immediate data (byte) into the pair register DE.

LHLI byte (Load HL with Immediate) .... SET A

1 Instruction code = 0 1 0 0 1 1 1 0 | $I_7 I_6 I_5 I_4 I_3 I_2 I_1 I_0$

2 Bytes = 2

3 Cycles = 2

4 Function = HL - byte byte = I_{7-0} = 00 - $FF_{(16)}$

This instruction loads 8 bits of immediate data (byte) into the pair register HL.
The LHLI instruction has a string effect. If LHLI or LHLT instruction appear in sequence, only the first one will be executed. The 2nd instruction and the remainder are executed as NOP instruction.

LHLI n5 (Load HL with Immediate) .... SET B

1 Instruction code = $\boxed{1\ 1\ 0\ I_4 I_3 I_2 I_1 I_0}$

2 Bytes = 1

3 Cycles = 1

4 Function = $H_0 L_{3-0}$ - n5, H_{3-1} - 000

This instruction loads 5 bit immediate data (n5) to the least significant bit (H_0) and L registers (L_{3-0}) of the HL register. The high-order 3 bits (H_{3-1}) of H register is unconditionally loaded with zero.

When LHLI instruction is placed continously because it has a stacking effect, then the instructions from the 2nd onward are processed as NOP.

LHLT taddr (Load HL with Table data) .... SET A only

1 Instruction code = 1 1 0 0 $P_3P_2P_1P_0$

2 Bytes = 1

3 Cycles = 2

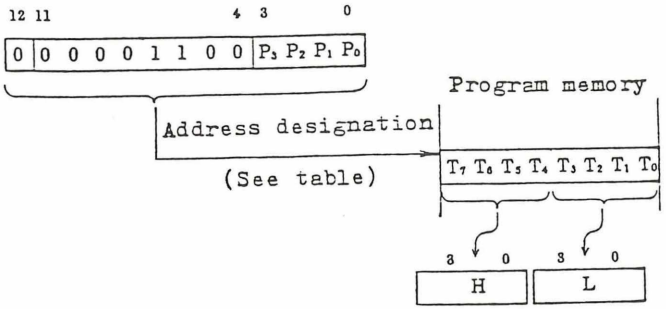
4 Function = HL - TABL (0 0 0 0 0 1 1 0 0 P_3 P_2 P_1 P_0)

This instruction loads high-order 4 bits of the program memory table data (T7-0) addressed by 4 bit immediate data (P_{3-0}) to the H register, and low-order 4 bits to the L register.

Since 00C0H - 00CFH address becomes the table area of the LHLT instruction, it is necessary to program table data with assembler directives in these areas in advance.

Bank flag (BNK) and program counter (PC) are not affected by the execution of this instruction.

But because it has a stacking effect, LHLT instruction or LHLI instruction, if continously placed, is processed as NOP after the 2nd one.



5 Example:

A subroutine example to load the table data into HL pair register.

```
TCAL1 :   LHLT   TABL1    ; (1)
TCAL2 :   LHLT   TABL2    ; (2)
TCAL3 :   LHLT   TABL3    ; (3)
          LHLI   00       ; (4)
          RT
           :
          ORG    0C0H

TABL1 :   DB     8CH      ; LHLT TABLE AREA
TABL2 :   DB     4BH
TABL3 :   DB     2AH
```

1 When TCAL1 is called
H register is loaded "8" and L register is loaded "C(H)", the instructions (2), (3), (4) are executed as NOP instructions. After that, RT instruction is executed.

2 When TCAL2 is called
H register is loaded "4", L register is loaded "B(H)" and the instructions (3), (4) are executed as NOP instruction. After that, RT instruction is executed.

3 When TCAL3 is called
H register is loaded "2", L register is loaded "A(H)" and the instruction (4) is executed as NOP instruction. After that, RT instruction is executed.

LAMT (Load A and Memory with Table data)

1 Instruction code = 0 1 0 1 1 1 1 0

2 Bytes = 1

3 Cycles = 2

4 Function = A - TABLE (BNK, PC_{11-6}, 0, C, A_{3-0})$_H$

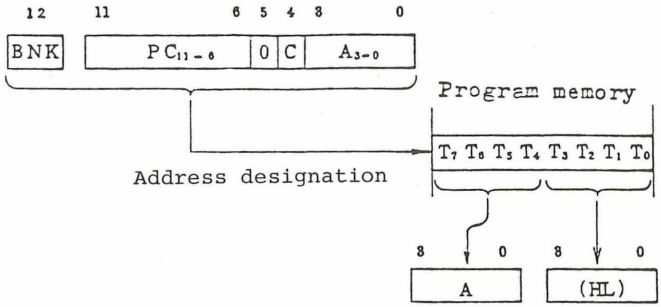
(HL) - TABLE (BNK, PC_{11-6}, 0, C, A_{3-0})$_L$

This instruction loads the high-order 4 bits(T_{7-4}) of the table data in the program memory addressed by the contents of bank flag (BNK), high-order 6 bits (PC_{11-6}), of program counter, carry flag (C), and the contents of accumulator (bit 5 of table address must be 0) to the accumulator, and low-order 4 bits (T_{3-0}) of the table data to the data memory addressed by the pair register HL.

The high-order 7 bits of the table address are determined by the contents of BNK and PC_{11-6} at the time of execution of this instruction, but low-order 5 bits (32 locations) of the table address can be set by the previous program.

In the table area, it is necessary to program necessary data in advance by assembler directives.

The bank flag (BNK) and program counter (PC_{11-0}) are not affected by this instruction.


12
11
6
5
4
3
0
BNK
PC11-6
0
C
A3-0
Program memory
T7 T6 T5 T4 T3 T2 T1 T0
Address designation
8
0
8
0
A
(HL)

LAMTL (Load A and Memory with Table data Long)

1 Instruction code = 0 0 1 1 1 1 1 1 | 0 0 1 1 0 1 0 0

2 Bytes = 2

3 Cycles = 3

4 Function = A - TABLE $(BNK, PC_{11-8}, A_{3-0}, (HL)_{3-0})_H$

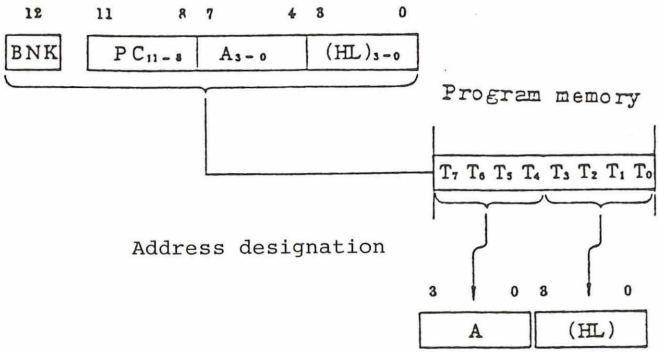
(HL) - TABLE $(BNK, PC_{11-8}, A_{3-0}, (HL)_{3-0})_L$

This instruction loads high-order 4 bits (T_{4-7}) of the table data in the program memory addressed by the contents of data memory $((HL)_{3-0})$, the content of bank flag (BNK), high-order 4 bits (PC_{11-8}) of the program counter, accumulator (A_{3-8}) to the accumulator, and low-order 4 bits (T_{3-0}) of the table data to the data memory addressed by the pair register HL.

The high-order 5 bits of table address are determined by the content of BNK and PC_{11-8} at the time of execution of this instruction, but the low-order 8 bits (245 location) can be set by a preceding program. As a result, the entire address space of program memory can be accessed.

In the table area, it is necessary to program necessary data in advance by assembler directives.

Bank flag (BNK) and program counter (PC_{11-0}) are not affected by execution of this instruction.



ST (Store A to Memory)

1 Instruction code = 0 1 0 1 0 1 1 1

2 Bytes = 1

3 Cycles = 1

4 Function = (HL) - A

The contents of the accumulator are stored in the memory addressed by the pair register HL.

STII n4 (Store Immediate to memory and Increment L) .... SET B only

1 Instruction code = $\boxed{0\ 1\ 0\ 0\ I_3\ I_2\ I_1\ I_0}$

2 Bytes = 1

3 Cycles = 1

4 Function = (HL) - n4, L - L + 1 n4 = I_{3-0} = 0 - FH

This instruction increments L register after storing 4 bit immediate data (n4) to the data memory addressed by pair register HL.

5 Example : Constants are stored in 10H - 13H of data memory.

```
LHLI   10H  ;  HL=10H
STII   C1   ;  (10H)=C1
STII   C2   ;  (11H)=C2
STII   C3   ;  (12H)=C3
STII   C4   ,  (13H)=C4
```

TAD (Transfer A to D) .... SET A only

1 Instruction code = $\boxed{0\ 0\ 1\ 1\ 1\ 1\ 1\ 0}$ $\boxed{1\ 0\ 1\ 0\ 1\ 0\ 1\ 0}$

2 Bytes = 2

3 Cycles = 2

4 Function = D - A

This instruction transfers the contents of the accumulator to the D register.

TAE (Transfer A to E) .... SET A only

1 Instruction code = 0 0 1 1 1 11 0 | 1 0 0 0 1 0 1 0

2 Bytes = 2

3 Cycles = 2

4 Function = E - A

This instruction transfers the contents of the accumulator to the E register.

TAH (Transfer A to H) .... SET A only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 1 1 1 0 1 0

2 Bytes = 2

3 Cycles = 2

4 Function = H - A

This instruction transfers the contents of the accumulator to the H register.

TAL (Transfer A to L) .... SET A only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 0 1 0

2 Bytes = 2

3 Cycles = 2

4 Function = L - A

This instruction transfers the contents of the accumulator to the L register.

TDA (Transfer D to A) .... SET A only

1 Instruction code = [0 0 1 1 1 1 1 0] [1 0 1 0 1 0 1 1]

2 Bytes = 2

3 Cycles = 2

4 Function = A - D

This instruction transfers the contents of the D register to the accumulator.

TEA (Transfer E to A) .... SET A only

1 Instruction code = [0 0 1 1 1 1 1 0] [1 0 0 0 1 0 1 1]

2 Bytes = 2

3 Cycles = 2

4 Function = A - E

This instruction transfers the contents of the E register to the accumulator.

THA (Transfer H to A) .... SET A only

1 Instruction code = [0 0 1 1 1 1 1 0] [1 0 1 1 1 0 1 1]

2 Bytes = 2

3 Cycles = 2

4 Function = A - H

This instruction transfers the contents of the H register to the accumulator.

TLA (Transfer L to A) .... SET A only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 0 1 1

2 Bytes = 2

3 Cycles = 2

4 Function = A - L

This instruction transfers the contents of the L register to the accumulator.

XAD (Exchange A with D) .... SET A only

1 Instruction code = 0 1 0 0 1 0 1 0

2 Bytes = 1

3 Cycles = 1

4 Function = A - D

This instruction exchanges the contents of the D register with the accumulator.

XAE (Exchange A with E) .... SET A only

1 Instruction code = 0 1 0 0 1 0 1 1

2 Bytes = 1

3 Cycles = 1

4 Function = A - E

This instruction exchanges contents of the D register with the accumulator.

XAH (Exchange A with H)

1 Instruction code = 0 1 1 1 1 0 1 0

2 Bytes = 1

3 Cycles = 1

4 Function = A - H

This instruction exchanges the contents of the H register with the accumulator.

XAL (Exchange A with L)

1 Instruction code = 0 1 1 1 1 0 1 1

2 Bytes = 1

3 Cycles = 1

4 Function = A - L

This instruction exchanges the contents of the L register with the accumulator.

XAM pr (Exchange A with Memory) .... SET A

1 Instruction code = $0\ 1\ 0\ R_4\ 0\ 1\ R_1 R_0$

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = A - (pr) pr = DL, DE, HL-, HL+, HL
When pr = HL+ = skip if L = 0
When pr = HL- = skip if L = $F_{(16)}$

R_2 R_1 R_0	pr
0 0 0	DL
0 0 1	DE
1 0 0	HL-
1 0 1	HL+
1 1 0	HL

This instruction exchanges the contents of the memory addressed by the pair register with the accumulator. When HL+ is specified with pr, the contents of the L register are increment after data exchange. So, when the result becomes 0, the succeeding one instruction is skipped. When HL- is specified with pr, the contents of the L register are decremented after data exchange. So, when the result becomes $F_{(16)}$, the succeeding one instruction is skipped.

XAM prl (Exchange A with Memory) ....SET B

1 Instruction code = 0 1 0 1 0 1 R_1 R_0

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = A - (prl) prl = HL-, HL+, HL
When prl = HL+, skip if L = 0
When prl = HL-, skip if L = FH

prl	R_1 R_0
HL-	0 0
HL+	0 1
HL	1 0

This instruction exchanges the contents of accumulator with the contents of data memory addressed by pair registers (HL-, HL+, HL). When HL+ is designated by prl, the contents of L register are incremented (+1) after data exchange. When the result becomes 0, the succeeding one instruction is skipped. When HL- is designated by prl, the contents of L register are decremented after data exchange, and when the result becomes FH (15), the succeeding one instruction is skipped.

5 Notes : XAM HL-, XAM HL+, XAM HL, can be written as XDS, XIS, and X (without operand in all these cases) respectively.

XADR mem (Exchange A with Memory Direct)

1 Instruction code = | 0 0 1 1 1 0 0 1 | | $D_7D_6D_5D_4D_3D_2D_1D_0$ |

2 Bytes = 2

3 Cycles = 2

4 Function = A - (mem) mem $D_7 - D_0 = 00 - FF_{(16)}$

This instruction exchanges the contents of the memory addressed by the 8 bits of immediate data (mem) with the accumulator.

5 Example: The contents of the data memory in the address location $2F_{(16)}$ are transferred to the address location $10_{(16)}$.

LADR 2FH = A - (2F)
XADR 10H = A - (10)

XHDR mem (Exchange H with Memory Direct)

1 Instruction code = | 0 0 1 1 1 0 1 0 | | $D_7D_6D_5D_4D_3D_2D_1D_0$ |

2 Bytes = 2

3 Cycles = 2

4 Function = H - (mem) mem = $D_7 - D_0 = 00 - FF_{(16)}$

This instruction exchanges the contents of the data memory addressed by 8 bits of immediate data (mem) with the H register.

XLDR mem (Exchange L with Memory Direct)

1 Instruction code = | 0 0 1 1 1 0 1 1 | | $D_7D_6D_5D_4D_3D_2D_1D_0$ |

2 Bytes = 2

3 Cycles = 2

4 Function = L - (mem) mem = $D_7 - D_0$ = 00 - $FF_{(16)}$

This instruction exchanges the contents of the data memory addressed by 8 bits of immediate data (mem) with the L register.

12.5.2 Arithmetic Instructions

AISC n4 (Add Immediate to A, Skip if Carry)

1 Instruction code = $\boxed{0\ 0\ 0\ 0\ I_3 I_2 I_1 I_0}$

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = A - A + n4 ; skip if carry
n4 = I_{3-0} = 1 - $F_{(16)}$

This instruction performs binary addition of 4 bit immediate data to the accumulator and if a carry is generated as a result of this addition, a skip is made. When n4 = 0, the instruction is executed as NOP instruction. The carry flag is not affected.

ASC (Add Memory to A, Skip if Carry)

1 Instruction code = $\boxed{0\ 1\ 1\ 1\ 1\ 1\ 0\ 1}$

2 Bytes = 1

3 Cycles = 1/2

4 Function = A - A + (HL) ; skip if carry.

This instruction performs binary addition of the contents of the memory addressed by the register HL to the accumulator, and if a carry is generated as a result of the addition, a skip is made. The carry flag is not affected.

ACSC (Add Memory to A with Carry, Skip if Carry)

1 Instruction code = 0 1 1 1 1 1 0 0

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = A, C - A + (HL) + C; skip if carry

This instruction performs binary addition of the contents of the memory addressed by the pair register HL, including the carry flag to the accumulator. If a carry is generated as a result of addition, the carry flag is set, and a skip is made. If a carry is not generated, the carry flag is reset. If the succeeding instruction is AISC, the skip condition (AISC instruction is skipped if a carry is generated) of the AISC instruction is prohibited. Instructions other than the AISC instruction will not be affected.

ADSC (Add D to A, Skip if Carry) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 1 0 1 0 0 1

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = A - A + D, Skip is carry

This instruction adds the contents of D register to the accumulator. If a carry is caused, a next instruction is skipped.

AESC (Add E to A, Skip if Carry) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 0 1 0 0 1

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = A - A + E, Skip if carry

This instruction adds the contents of the E register to the accumulator. If a carry is caused as a result of the addition, a next instruction is skipped.

AHSC (Add H to A, Skip if Carry) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 1 1 1 0 0 1

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = A - A + H, Skip if carry

This instruction adds the contents of the H register to the accumulator. If a carry is generated as a result of addition, a next instruction is skipped.

ALSC (Add L to A, Skip if Carry) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 0 0 1

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = A - A + L, Skip if carry

This instruction adds the contents of the L register to the accumulator. If a carry is generated as a result of addition, a next instruction is skipped.

SDSB (Subtract D from A, Skip if Borrow) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 1 0 1 0 0 0

2 Bytes = 2

3 Cycles = 2/3

4 Function = A - A - D, Skip if borrow

This instruction subtracts the contents of D register from the accumulator and stores the result to the accumulator. If a borrow occurs as a result of subtraction, a next instruction is skipped.

SESB (Subtract E from A, Skip of Borrow) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 0 1 0 0 0

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = A - A - E, Skip if borrow

This instruction subtracts the contents of E register from the accumulator and stores the result to the accumulator. If a borrow is caused as a result of subtraction, a next instruction is skipped.

SHSB (Subtract H from A, Skip if Borrow) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 1 1 1 0 0 0

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = A - A - H, Skip if borrow

This instruction subtracts the contents of the H register from the accumulator and stores the result to the accumulator. If a borrow is caused as a result of subtraction, a next instruction is skipped.

SLSB (Subtract L from A, Skip if Borrow) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 0 0 0

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = A - A - L, Skip if borrow

This instruction subtracts the contents of the L register from the contents of the accumulator and stores the result to the accumulator. If a borrow is caused as a result of subtraction, a next instruction is skipped.

EXL (Exclusive Or Logic A and Memory)

1 Instruction code = 0 1 1 1 1 1 1 0

2 Bytes = 1

3 Cycles = 1

4 Function = A - A ∨ (HL)

This instruction causes the contents of the memory addressed by the pair register HL to be performed exclusive OR with the contents of the accumulator and loads the result to the accumulator.

ANL (And Logic A and Memory)

1 Instruction code = 0 0 1 1 1 1 1 1 | 1 0 1 1 0 0 1 0

2 Bytes = 2

3 Cycles = 2

4 Function = A - A ∧ (HL)

This instruction causes the contents of the memory addressed by the contents of the HL register to be performed AND with the accumulator and loads the result to the accumulator.

ORL (OR Logic A and Memory)

1 Instruction code = | 0 0 1 1 1 1 1 1 | | 1 0 1 1 0 1 1 0 |

2 Bytes = 2

3 Cycles = 2

4 Function = A - A ∨ (HL)

This instruction causes the contents of the memory addressed by the contents of the HL register to be performed OR with the accumulator and loads the result to the accumulator.

12.5.3 Accumulator Operation Instructions

CMA (Complement A)

1 Instruction code = 0 1 1 1 1 1 1 1

2 Bytes = 1

3 Cycles = 1

4 Function = $A - \overline{A}$

This instruction performs one's compliment of the accumulator (each bit is inverted).

5 Example : Case of A = 5

	Before execution	3 2 1 0 0 1 0 1	A
CMA			
	After execution	1 0 1 0	A

RAR (Rotate A Right)

1 Instruction code = 0 0 1 1 1 1 1 1 | 1 0 1 1 0 0 1 1

2 Bytes = 2

3 Cycles = 2

4 Function = An-1 - An (n = 1 to 3)

$C - A_0$

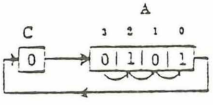
$A_3 - C$

This instruction rotates the contents of the accumulator including the carry flag to the right bit by bit.

5 Example :

RAR

Before execution



After execution

1 | 0 0 1 0

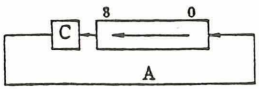
RAL (Rotate A Left)

1 Instruction code = | 0 0 1 1 1 1 1 1 | | 1 0 1 1 0 1 1 1 |

2 Bytes = 2

3 Cycles = 2

4 Function = C - A_3, An - An-1, A_0 C n=1 - 3



This instruction rotates the contents of the accumulator including the carry flag to the left bit by bit.

12.5.4 Carry Flag Operation Instructions

RC (Reset Carry)

1 Instruction code = 0 1 1 1 1 0 0 0

2 Bytes = 1

3 Cycles = 1

4 Function = C - 0

This instruction resets the carry flag.

SC (Set Carry)

1 Instruction code = 0 1 1 1 1 0 0 1

2 Bytes = 1

3 Cycles = 1

4 Function = C - 1

This instruction sets the carry flag.

12.5.5 Increment/Decrement Instructions

IES (Increment E, Skip if E = 0) .... SET A only

1 Instruction code = 0 1 0 0 1 0 0 1

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = E - E + 1 ; skip if E = 0

This instruction increments the E register and a next instruction is skipped if the result of the increment becomes E = 0.

ILS (Increment L, Skip if L = 0)

1 Instruction code = 0 1 0 1 1 0 0 1

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = L - L + 1 ; skip if L = 0

This instruction increments the L register, and a next instruction is skipped if the result of the increment becomes L = 0.

IDE (Increment DE) .... SET A only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 0 1 1 0 1

2 Bytes = 2

3 Cycles = 2

4 Function = DE - DE + 1

This instruction increments the pair register DE as an 8 bit register.

IHL (Increment HL) .... SET A only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 1 0 1

2 Bytes = 2

3 Cycles = 2

4 Function = HL - HL + 1

This instruction increments (+1) the pair register HL as an 8 bit register.

IDRS mem (Increment Memory Direct, Skip if Memory = 0)

1 Instruction code = 0 0 1 1 1 1 0 1 | $D_7 D_6 D_5 D_4 D_3 D_2 D_1 D_0$

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = (mem) - (mem) + 1
Skip if (mem) = 0 mem = $D_7 - D_0$ = 00 - $FF_{(16)}$

This instruction increments the contents of the memory addressed by 8 bit immediate data (mem), and a next instruction is skipped if the result of the increment becomes 0.

DES (Decrement E, Skip if E = F) .... SET A only

1 Instruction code = 0 1 0 0 1 0 0 0

2 Bytes = 1

3 Cycles = 1/2

4 Function = E - E - 1 Skip if E = $F_{(16)}$

This instruction decrements the L register, and a next instruction is skipped if the result of the decrement becomes L = $F_{(16)}$.

DLS (Decrement L, Skip if L = F)

1 Instruction code = 0 1 0 1 1 0 0 0

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = L - L - 1; skip if L = $F_{(16)}$

This instruction increments the L register, and a next instruction is skipped if the result of the decrement becomes L = $F_{(16)}$.

DDE (Decrement DE) .... SET A only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 0 1 1 0 0

2 Bytes = 2

3 Cycles = 2

4 Function = DE - DE - 1

This instruction decrements (-1) the pair register DE as an 8 bit register.

DHL (Decrement HL) .... SET A only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 1 0 0

2 Bytes = 2

3 Cycles = 2

4 Function = HL - HL - 1

This instruction decrements (-1) the pair register HL as an 8 bit register.

DDRS mem (Decrement Memory Direct, Skip if Memory = F)

1 Instruction code = | 0 0 1 1 1 1 0 0 | $D_7D_6D_5D_4D_3D_2D_1D_0$ |

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = (mem) - (mem) - 1
Skip if (mem) = $F_{(16)}$ mem = D_{7-0} = 00 - $FF_{(16)}$

This instruction decrements the contents of the memory addressed by 8 bit immediate data mem, and a next instruction is skipped if the result of the decrement become $F_{(16)}$.

12.5.6 Memory Bit Operation Instructions

RMB Bit (Reset Memory Bit)

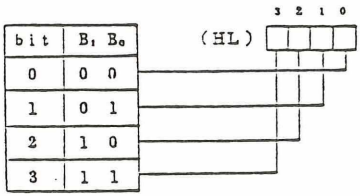
1 Instruction code = 0 1 1 0 1 0 $B_1 B_0$

2 Bytes = 1

3 Cycles = 1

4 Function = (HL bit - 0 bit = B_{1-0}; 0 - 3

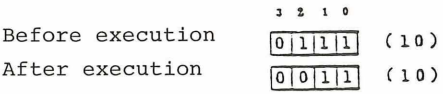
This instruction resets the bit specified by 2 bit immediate data (bit) of the memory addressed by the pair register HL.



Those other than the specified bit are not changed.

5 Example : Bit 2 of data memory (10) is reset.

```
LHLI   10H
RMB    2      : (10)3 - 0
```



SMB Bit (Set Memory Bit)

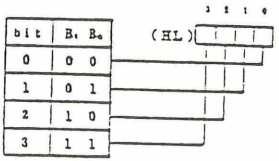
1 Instruction code = | 0 1 1 0 1 1 B_1 B_0 |

2 Bytes = 1

3 Cycles = 1

4 Function = (HL) bit - 1 bit = B_{1-0} ; 0 - 3

This instruction sets the bit specified by 2 bit immediate data (bit) of the memory addressed by the pair register.



Those other than the specified bit are not changed.

5 Example : Bit 1 of the data memory (11) is set.

```
LHLI    11H
SMB     1       :(11)3  1
```

Before execution | 0 | 0 | 0 | 0 | (11)

After execution | 0 | 0 | 1 | 0 | (11)

12.5.7 Jump Instruction

JMP addr1 (Jump) .... SET A

1 Instruction code = 0 0 1 0 $P_{11}P_{10}P_9P_8$ $P_7P_6P_5P_4P_3P_2P_1P_0$

2 Bytes = 2

3 Cycles = 2

4 Function = P C_{11} - P_{11-0} addr 1=BNK,P_{11-0}:000H-1FFFH

This instruction causes a jump to the address location addressed by the content of bank flag (BNK) and 12 bits immediate data (P_{11-0}). BNK does not change, however. A jump can be made to the entire address space within the bank written in the instruction.

JMP addr2 (Jump) .... SET B

1 Instruction code = 0 0 1 0 0 $P_{10}P_9P_8$ $P_7P_6P_5P_4P_3P_2P_1P_0$

2 Bytes = 2

3 Cycles = 2

4 Function = PC_{11-0} - 0, P_{10-0}

addr2=BNK, 0, P_{10-0} : 0000H-07FFH
1000H-17FFH

This instruction causes a jump to the address location addressed by the contents of bank flag (BNK) and 11 bit immediate data (P_{10-0}). BNK does not change, however. A jump can be made to the first half address space within the bank written in this instruction.

JMP addr (Jump Long)

1 Instruction code = 0 0 1 1 1 1 1 1 | 0 0 P_{13} 0 P_{11} P_{10} P_9 P_8

2 Bytes = 3 | P_7 P_6 P_5 P_4 P_3 P_2 P_1 P_0

3 Cycles = 3

4 Function = BNK - P_{13}, PC_{11-0} - P_{11-0}

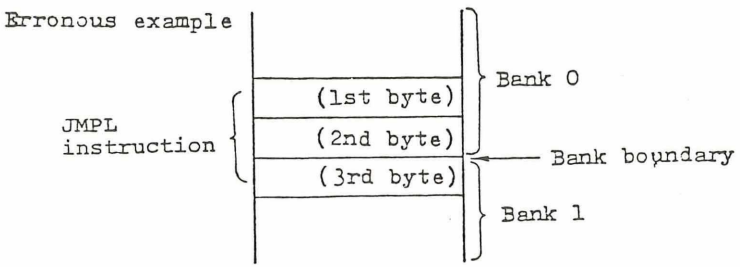
addr=$P_{13}P_{11-0}$: 0000H-1FFFH

This instruction causes a jump to the address location addressed by 12 bit immediate data (P_{11-0}) within the bank designated by immediate data (P_{13}). A jump can be made to the desired address in the desired bank.

5 Notes :

(1) This instruction cannot be put immediately after an instruction with a skip function because of being a 3 bytes instruction. Therefore, the JMPL instruction cannot be skipped.

(2) It is possible to put the JMPL instruction on the bank boundary of the bank 0 and bank 1. This is reason why the program counter and bank flag operate independently of each other.



JCP addr (Jump in the Current Page)

1 Instruction code = $\boxed{1\ 0\ P_5P_4P_3P_2P_1P_0}$

2 Bytes = 1

3 Cycles = 1

4 Function = $PC_{5-0} \leftarrow P_{5-0}$

addr=BNK, $PC_{11-6} \cdot P_{5-0}$: 0000H-1FFFH

This instruction causes a jump to the address location replacing the low-order 6 bits (PC_{5-0}) of the program counter with the 6 bits immediate data (P_{5-0}). The instruction does not give any effects on the bank flag (BNK) and high-order 6 bits (PC_{11-6}) of the progam counter.

"addr" can be written the values different from the current BNK and PC_{11-6} contents.

JAM addr 3 (Jump with A and Memory)

1 Instruction code = | 0 0 1 1 1 1 1 1 | | 0 0 0 1 $P_3P_2P_1P_0$ |

2 Bytes = 2

3 Cycles = 2

4 Function = PC_{11-8} - P_{3-0}, PC_{7-4} - A , PC_{3-0} - (HL)

addr3=P_{3-0} : 0 - FH

This instruction causes a jump to the address location replacing the high-order 4 bits (PC_{11-8}) of program counter with the 4 bits immediate data (addr3), the midle-order 4 bits (PC_{7-4}) of program counter with the contents of accumulator and the low-order 4 bits (PC_{3-0}) of program counter with the contents of the data memory addressed by the pair register HL. BNK doesn't change however.
A jump can be made to the entire address space within the bank in which the JAM instruction is placed.

12.5.8 Subroutine Stack Control Instructions

CALL caddr (Call subroutine)

1 Instruction code = $\boxed{0\ 0\ 1\ 1\ 0\ P_{10}P_9P_8}$ $\boxed{P_7P_6P_5P_4P_3P_2P_1P_0}$

2 Bytes = 2

3 Cycles = 2

4 Function = (SP-1) - PC_{7-4}, (SP-2) - PC_{3-0}, (SP-3) - PSW,
(SP-4) - PC_{11-8}, SP - SP-4
BNK - 0, PC_{11-0} - 0, P_{10-0}
caddr=0,0,P_{10-0} : 0000H-07FFH

This instruction causes a jump to the address location addressed by the 11 bit immediate data (caddr) after storing the contents of program counter (PC_{11-0}: return address) and the contents of PSW in the data memory (stack) addressed by the stack pointer (SP).

The range in which it can be called is limited to 000H-7FFH (0 - 2047) address within the bank 0.

CAL caddrl (Call subroutine) .... SET B only

1 Instruction code = $\boxed{1\ 1\ 1\ P_4P_3P_2P_1P_0}$

2 Bytes = 1

3 Cycles = 2

4 Function = (SP-1) - PC_{7-4}, (SP-2) - PC_{3-0}, (SP-3) - PSW

(SP-4) - PC_{11-8}, SP - SP-4

BNK - 0, PC_{11-0} - $0001P_4P_3000P_2P_1P_0$

caddrl=0, $0001P_4P_3000P_2P_1P_0$:0100H-0107H

0140H-0147H

0180H-0187H

01C0h-01C7H

This instruction causes a jump to the address location determined by 5 bit immediate data (P_{4-0}) after storing the contents of program counters (PC_{11-0}: return address) and the contents of PSW (including BNK) in the data memory (stack) to be addressed by the stack pointer (SP). The range in which this instruction can be called is limited to the specific area of bank 0.

CALT taddrl (Call with Table data) .... SET A Only

1 Instruction code = $\boxed{1\ 1\ P_5P_4P_3P_2P_1P_0}$

2 Bytes = 1

3 Cycles = 2

4 Function = (SP-1) - PC_{7-1}, (SP-2) - PC_{3-0}, (SP-3) - PSW

(SP-4) - PC_{11-8}, SP - SP-4

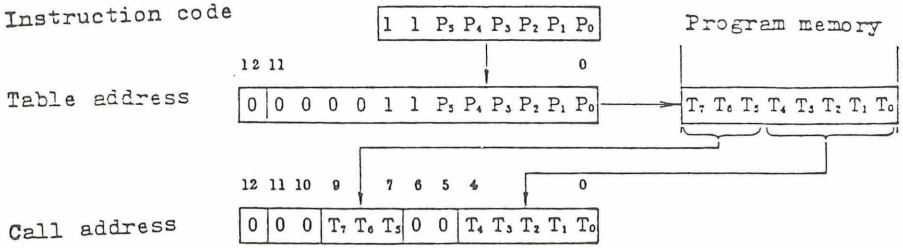
BNK - 0, $PC_{11,10,6,5}$ - 0,

$PC_{9-7,\ 4-0}$ - TABLE(0000011$P_5P_4P_3P_2P_1P_0$)

taddrl = 0000011$P_5P_4P_3P_2P_1P_0$: 00D0H - 00FFH

This instruction loads table data (T_{7-0}), to be referred to by taddrl, in the form of 0 0 T_7 T_6 T_5 0 0 T_4 T_3 T_2 T_1 T_0 to the program counter (PC_{11-0}) after storing the program counters (PC_{11-0}: return address) and the contents of program status word (PSW; including BNK) to the data memory (stack) addressed by the stack pointer (SP). It thencauses these to be jumped. The address range in which the table can be provided is 0D0-0FFH of bank 0 (208 - 255 address). The range that can be called is the specific address of bank 0 as follows.

000H - 01FH (0 - 31)
080H - 09FH (128 - 169)
100H - 11FH (256 - 287)
180H - 19FH (384 - 415)
200H - 21FH (512 - 543)
280H - 29FH (640 - 671)
300H - 31FH (768 - 799)
380H - 39FH (896 - 927)


Instruction code
1 1 P5 P4 P3 P2 P1 P0
Program memory
12 11
0
Table address
0 0 0 0 0 1 1 P5 P4 P3 P2 P1 P0
T7 T6 T5 T4 T3 T2 T1 T0
12 11 10 9
7 6 5 4
0
Call address
0 0 0 T7 T6 T5 0 0 T4 T3 T2 T1 T0

6 Notes:

The operand taddrl of the CALT instruction is not subroutine call address. So, table address has to be described. In the table area (00D0H - 00FFH), it is necessary to program call address data (T_{7-0}) in advance by the assembler directives (DET).

RT (Return from subroutine)

1 Instruction code = 0 1 0 1 0 0 1 1

2 Bytes = 1

3 Cycles = 2

4 Function = PC_{11-6} - (SP), BNK - $(SP+1)_1$, PC_{3-0} - (SP+2)
PC_{7-4} - (SP+3), SP - SP+4

This instruction restores the contents of data memory (stack) addressed by the stack pointer (SP) to the program counters (PC_{11-0}) and the bank flag (BNK), and then increments the contents of SP.

6 Notes: Three bits of the program status words (PSW), namely, SK1, SK0, and C, cannot be restored.

RTS (Return from subroutine, then skip unconditionally)

1 Instruction code =

2 Bytes = 1

3 Cycles = 3

4 Function = PC_{11-8} - (SP), BNK - $(SP+1)_1$, PC_{3-0} - (SP+2),
PC_{7-4} - (SP+3), SP+4,
Then skip unconditionally

This instruction restores the contents of data memory (stack) addressed by the stack pointer (SP) to the program counters (PC_{11-0}) and bank flag (BNK) and then increments SP, after that, the next instruction is unconditionally skipped.

6 Notes: Three bits of the program status words (PSW) namely, SK1, SK0 and C cannot be restored.

RTPSW (Return from subroutine and restore PSW) .... SET A Only

1 Instruction code = 0 1 0 0 0 0 1 1

2 Bytes = 1

3 Cycles = 2

4 Function = PC_{11-8} - (SP), PSW - (SP+1), PC_{3-0} - (SP+2)
PC_{7-4} - (SP+3), SP - SP+4

This instruction restores the contents of data memory (stack) addressed by the stack pointer (SP) to the program counters (PC_{11-0}) and program status words (PSW) including BNK and then increments SP.

This instruction is used in case the interrupt routine is returned or the restoring PSW is required.

PSHDE (Push DE Register on Stack) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 0 1 1 1 0

2 Bytes = 2

3 Cycles = 2

4 Function = (SP-1) - D, (SP-2) - E
SP - SP-2

This instruction stores the contents of the D register and those of the E register to the data memory (stack) addressed by the stack pointer (SP) and then decrements the SP.

PSHHL (Push HL Register on Stack) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 1 1 0

2 Bytes = 2

3 Cycles = 2

4 Function = (SP-1) - H, (SP-2) - L
SP - SP-2

This instruction stores the contents of the H register and those of the L register to the data memory (stack) addressed by the stack pointer (SP) and then decrements the SP.

POPDE (Pop DE Register off Stack) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 0 1 1 1 1

2 Bytes = 2

3 Cycles = 2

4 Function = E - (SP), D - (SP+1)
SP - SP+2

This instruction restores the contents of the data memory (stack) addressed by the stack pointer (SP) to the D register and the E register and then increments the SP.

POPHL (Pop HL Register off Stack) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 1 0 0 1 1 1 1 1

2 Bytes = 2

3 Cycles = 2

4 Function = L - (SP), H - (SP+1)
SP - SP+2

This instruction restores the contents of the data memory (stack)addressed by the stack pointer to the H register and the L register and then increments the SP.

TAMSP (Transfer A and Memory to SP)

1 Instruction code = | 0 0 1 1 1 1 1 1 | | 0 0 1 1 0 0 0 1 |

2 Bytes = 2

3 Cycles = 2

4 Function = SP_{7-4} - A, SP_{3-1} - $(HL)_{3-1}$, SP_0 - 0

This instruction transfers the contents of the accumulator to the high-order 4 bits (SP_{7-4}) of the stack pointer, and the high-order 3 bits of the contents of the data memory addressed by the HL register to low-order 3 bits (SP_{3-1}).

TSPAM (transfer SP to A and Memory)

1 Instruction code = | 0 0 1 1 1 1 1 1 | | 0 0 1 1 0 1 0 1 |

2 Bytes = 2

3 Cycles = 2

4 Function = A - SP_{7-4}, $(HL)_{3-1}$ - SP_{3-1}, $(HL)_0$ - 0

This instruction transfers the high-order 4 bits (SP_{7-4}) of the stack pointer to the accumulator, the low-order 3 bits (SP_{3-1}) of the stack pointer to the high-order 3 bits of data memory addressed by HL register. The least significant bit of the data memory is automatically transferred 0.

12.5.9 Skip Instructions

S C (Skip if Carry)

1 Instruction code = 0 1 0 1 1 0 1 0

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = Skip if C = 1

This instruction causes a skip to be performed if the carry flag is 1.

SKMBT bit (Skip if Memory Bit True)

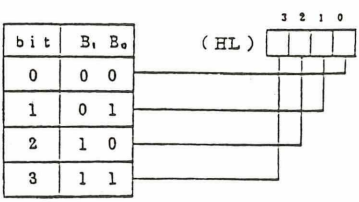
1 Instruction code = 0 1 1 0 0 B_1 B_0

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = Skip if(HL) bit = 1 bit = B_{1-0} : 0 - 8

This instruction causes a skip to be performed if the bit specified by the 2 bits immediate data of the memory addressed by the pair register HL is 1.



bit	B_1 B_0
0	0 0
1	0 1
2	1 0
3	1 1

SKABT bit (Skip if A Bit True)

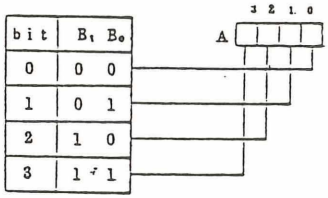
1 Instruction code = 0 1 1 1 0 1 B_1B_o

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = Skip if A bit = 1

This instruction causes a skip to be performed if the bit specified by the 2 bits immediate data of the accumulator is 1.



SXMBF bit (Skip if Memory Bits False)

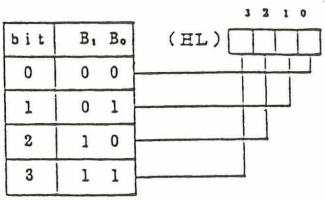
1 Instruction code = 0 1 1 0 0 0 B_1B_o

2 Bytes = 1

3 Cycles = 1/2 (when skipping)

4 Function = Skip if (HL) bit = 0 bit = B_{1-0} : 0 - 8

This instruction causes a skip to be performed if the bit specified by 2 bits immediate data of the memory addressed by the pair register HL is 0.



5 Example: Test is made to determine whether or not the bit 3 of the data memory (10) is 0.

```
        LHLI   10H : HL = 10
        SKMBF  3   : Skip if (10)3 = 0
        JMP    ERR : JUMP TO ERR
    OK:
```

SKAEM (skip if A Equals Memory)

1 Instruction code = 0 1 0 1 1 1 1 1
2 Bytes = 1
3 Cycles = 1/2 (when skipping)
4 Function = Skip if A = (HL)

This instruction causes a skip to be performed if the contents of the accumulator equal the contents of the data memory addressed by the pair register HL.

5 Example: The number of data 5 existing in the data memory (10 - 1F) are recorded in the data memory (20).

```
COMP  :  LAI 0
         XADR 20H : CLEAR COUNTER
         LHL1 1FH : HL = 1F
         LAI  5
LOOP  :  SKAEM    : SKIP IF (HL) = 5
         JCP  BR
         IDRS 20H : INCREMENT COUNTER
BR    :  DLS      : DECREMENT DATA POINTER
         JCP  LOOP
END   :
```

First, the coincidence counter (20H) is cleared, 5 is set to the accumulator, and the contents of the data memory are compared with those of the accumulator by the SKAEM instruction while updating the address of data memory, to increment the coincidence counter if those are in coincidence.

SKAEI n4 (Skip if A Equals Immediate)

1 Instruction code = 0 0 1 1 11 1 1 | 0 1 1 0 $I_3I_2I_1I_0$

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = Skip if A = n4 n4 = I_{3-0} : 0 - $F_{(16)}$

This instruction causes a skip to be performed if 4 bits immediate data equal the contents of the accumulator.

SKDEI n4 (Skip if D Equals Immediate) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 0 | 0 1 1 0 $I_3I_2I_1I_0$

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = Skip if D = n4 n4 = I_{3-0} : 0 - $F_{(16)}$

This instruction causes a skip to be performed if 4 bits immediate data n4 equals the contents of the D register.

SKEEI n4 (Skip if E Equals Immediate) .... SET A Only

1 Instruction code = [0 0 1 1 1 1 1 0] [0 1 0 1 $I_3I_2I_1I_0$]

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = Skip if E = n4 n4 = I_{3-0} : 0 - $F_{(16)}$

This instruction causes a skip to be performed if 4 bits immediate data n4 equals the contents of the E register.

SKHEI n4 (Skip if H Equals Immediate) .... SET A Only

1 Instruction code = [0 0 1 1 1 1 1 0] [0 1 1 1 $I_3I_2I_1I_0$]

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = Skip if H = n4 n4 = I_{3-0} : 0 - $F_{(16)}$

This instruction causes a skip to be performed if 4 bits immediate data n4 equals the contents of the H register.

SKLEI n4 (Skip if L Equals Immediate) .... SET A

1 Instruction code = [0 0 1 1 1 1 1 0] [0 1 0 1 $I_3I_2I_1I_0$]

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = Skip if L = n4 n4 = I_{3-0} : 0 - $F_{(16)}$

This instruction causes a skip to be performed if 4 bits immediate data n4 equals the contents of the L register.

SKLEI n4 (Skip if L Equals Immediate) .... SET B

1 Instruction code = 0 0 1 1 1 1 1 1 | 0 1 0 1 $I_3I_2I_1I_0$

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = Skip if L = n4 n4 = I_{3-0} : 0 - FH

This instruction causes a skip if the contents of L register equal 4 bits immediate data (n4).

SKMEI n4 (Skip if Memory Equals Immediate)

1 Instruction code = 0 0 1 1 1 1 1 1 | 0 1 1 1 $I_3I_2I_1I_0$

2 Bytes = 2

3 Cycles = 2/3 (when skipping)

4 Function = Skip if (HL) = n4 n4 = I_{3-0} : 0 - FH

This instruction causes a skip if the contents of data memory addressed by the pair register HL equal 4 bits immediate data (n4).

12.5.10 SIO Control Instructions

TAMSIO (transfer A and Memory to SIO)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 1 1 1 0]

2 Bytes = 2

3 Cycles = 2

4 Function = SIO_{7-4} - A, SIO_{3-0} - (HL)

This instruction transfers the contents of the accumulator to the high-order 4 bits (SIO_{7-4}) of the shift register, and the contents of data memory addressed by the HL register to the low-order 4 bits (SIO_{3-0}) of the shift register.

TSIOAM (transfer SIO to A and Memory)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 1 0 1 0]

2 Bytes = 2

3 Cycles = 2

4 Function = A - SIO_{7-4} , (HL) - SIO_{3-0}

This instruction transfers the high-order 4 bits (SIO_{7-4}) of the shift register to the accumulator and the low-order 4 bits (SIO_{3-0}) of the shift register to the data memory addressed by the HL register.

SIO (Start SIO)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 0 0 1 1]

2 Bytes = 2

3 Cycles = 2

4 Function = 3 BIT CNT - 0 , INT 0/S RQF - 0

This instruction clears the 3 bits counter of the serial interface and resets the INT O/S request flag and shift operation has been started.

12.5.11 Timer Control Instructions

TAMMOD (transfer A and Memory to Timer Modulo Register)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 1 1 1 1]
2 Bytes = 2
3 Cycles = 2
4 Function = MOD_{7-4} - A , MOD_{3-0} - (HL)

This instruction transfers the contents of the accumulator to the high-order 4 bits (MOD_{7-4}) of the modulo register, and the contents of the data memory addressed by the HL register to the low-order 4 bits (MOD_{3-0}) of modulo register.

Timer (Start Timer)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 0 0 1 0]
2 Bytes = 2
3 Cycles = 2
4 Function = CT - 0 , INT T RQF - 0

This instruction clears the count register (CT) and resets the INT T request flag, and timer operation has been started.

TCNTAM (Transfer Timer Count Register to A and Memory)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 1 0 1 1]
2 Bytes = 2
3 Cycles = 2
4 Function = A - CT_{7-4} , (HL) - CT_{3-0}

This instruction transfers the high-order 4 bits (CT_{7-4}) of the count register to the accumulator and the low-order 4 bits (CT_{3-0}) to the data memory addressed by the HL register.
Count operation is halted during execution of this instruction

12.5.12 Interrupt Control Instructions

EI n4 (Enable Interrupt) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 1 | 1 0 0 1 $I_3 I_2 I_1 I_0$

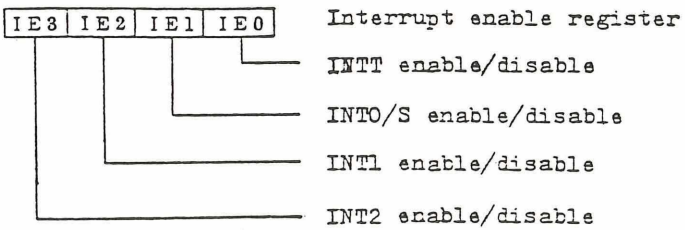
2 Bytes = 2

3 Cycles = 2

4 Function = IME F/F - 1 (n4=0)

IE3-0 - IE3-0 - n4 (n4≠0)

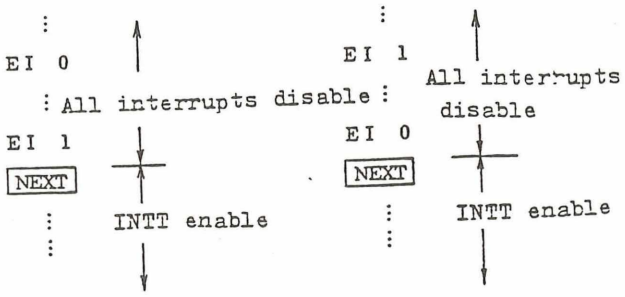
$n4=I_{3-0}$: 0 -FH



When 4 bits immediate data n4 = 0, this instruction sets interrupt master enable flip-flop (IME F/F) high level (1). When n4 = 0, it causes n4 to be ORed with the interrupt enable registers (IE3-0) and the result is stored in IE3-0.

When IME F/F = 1 and that specialbit of IE3-0 corresponded with that special interrupt source is 1, the interrupt source becomes enable state. But the enable state is actually established from the next instruction cycle of EI instruction.

5 Example : When started with IME F/F = 0, and
IE3-0 = 0000.



Example 1 Example 2

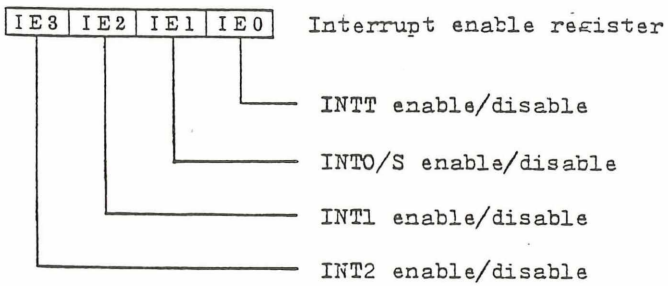
DI n4 (Disable Interrupt) .... SET A Only

1 Instruction code = 0 0 1 1 1 1 1 1 | 1 0 0 0 I_3 I_2 I_1 I_0

2 Bytes = 2

3 Cycles = 2

4 Function = IME F/F - 0 (n4=0)
IE3-0 - IE3 - 0 ∧ n4 (n4=0)
$n4=I_{3-0}$: 0 - FH

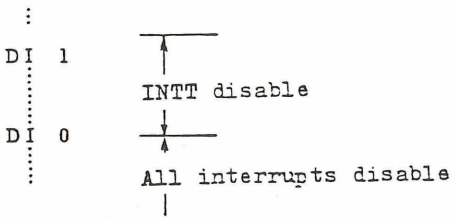


This instruction sets the interrupt master enable (IME F/F) low level 0 when 4 bit immediate data n4=4, thereby disabling all the interrupts.

When n4 $\neq$ 0, the instruction causes the interrupt enable registers (IE3-0) to be ANDed with n4's complement ($\overline{n4}$) and the result is stored in IE3-0.

When the each bit of IE3-0 corresponded with the each interrupt source is 0, the interrupt source is placed in the interrupt disable state. Interrupt disable state is established during the execution of DI instruction.

5 Example : When started with IME F/F = 1, IE3-0 = 111.



SKI n4 (Skip if Interrupt flag is true)

1 Instruction code = 0 0 1 1 1 1 1 1 | 0 1 0 0 $I_3I_2I_1I_0$

2 Bytes = 2

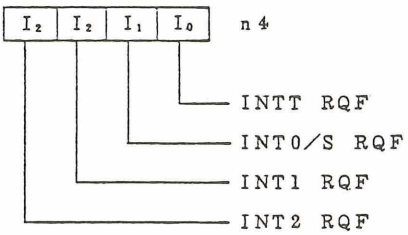
3 Cycles = 2/3 (when skipping)

4 Function = Skip if INT RQF $\wedge$ n4 = 0

INT RQF - INT RQF $\wedge \overline{n4}$

n4=I_{3-0} : 1 - FH

This instruction skips if the interrupt request flag (INTX RQF) corresponding to the bit which has set 1 by 4 bits immediate data (n4) is 1, and then resets the corresponding flag. The n4 and the corresponding bits of request flag are as below.



⑤ Example : When testing INT2 RQF.

```
SKI  8     ; I3I2I1I0=1000
```

12.5.13 Input/Output Instructions

IPL (Input Port specified by L)

1 Instruction code = 0 1 1 1 0 0 0 0
2 Bytes = 1
3 Cycles = 1
4 Function = A - Port (L)

Contents of L Register	0	1	4	5	6	7	8	9	AH	BH
Port Name	Port 0	Port 1	Port 4	Port 5	Port 6	ポート7	8243 Port 4	8243 Port 5	8243 Port 6	8243 Port 7

When L = 0, 1, 4-7, the contents of the port spcified by L register are loaded in the accumulator.

When L = 8-BH, the contents of expansion port of 8243 I/O expander is loaded in the accumulator via port 1 of μPD7500.

IP addr5 (Input Port specified by immediate)

1 Instruction code = 0 0 1 1 1 1 1 1 | 1 1 0 0 $D_3D_2D_1D_0$
2 Bytes = 2
3 Cycles = 2
4 Function = A - Port (addr5) addr5=D_{3-0} : 0,1,4-BH

addr5	0	1	4	5	6	7	8	9	AH	BH
Port Name	Port 0	Port 1	Port 4	Port 5	Port 6	Port 7	8243 Port 4	8243 Port 5	8243 Port 6	8243 Port 7

When addr5 = 0, 1, 4-7, the contents of the port designated by 4 bits immediate data (addr5) are loaded in the accumulator.
If addrr5 = 8-BH, the contents of the expansion port of the 8243 I/O expander designated by addr5 are loaded in the accumulator via port 1 of μPD7500.

IP1 (Input Port 1)

1 Instruction code = 0 1 1 1 0 0 0 1
2 Bytes = 1
3 Cycles = 1
4 Function = A - Port 1

This instruction loads the contents of Port 1 in the accumulator.

IP54 (Input Port 5 and Port 4)

1 Instruction code = 0 0 1 1 1 1 1 1 | 0 0 1 1 1 0 0 0
2 Bytes = 2
3 Cycles = 2
4 Function = A - Port 5, (HL) - Port 4

This instruction loads the contents of port 5 in the accumulator and loads the contents of the port 4 in the data memory addressed by the HL register.

OPL (Output Port specified by L)

1 Instruction code = 0 1 1 1 0 0 1 0
2 Bytes = 1
3 Cycles = 1
4 Function = Port/Mode register (L) - A

Contents of L Register	1	2	3	4	5	6	7	8	9	AH	BH
Port Name	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7	8243 Port 4	8243 Port 5	8243 Port 6	8243 Port 7

Contents of L Register	CH	DH	EH	FH
Mode Register Name	Clock mode register	INT1 mode register	Port 6 mode register	Shift mode register

When L = 1-7, the contents of accumulator are output to the port specified by the contents of L register.

If L = 8-BH, the instruction causes the contents of accumulator to be output to the expansion port of 8243 I/O expander specified by L register via port 1.

If L = CH-FH, the contents of accumulator are transferred to the mode register specified by L register.

OP addr4 (Output Port specified by immediate)

1 Instruction code = [0 0 1 1 1 1 1 1] [1 1 1 0 $D_3D_2D_1D_0$]

2 Bytes = 2

3 Cycles = 2

4 Function = Port/Mode register (addr4) - A

addr4=D_{3-0} : 1-FH

addr4	1	2	3	4	5	6	7	8	9	AH	BH
Port Name	Port 1	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7	8243 Port 4	8243 Port 5	8243 Port 6	8243 Port 7

addr4	CH	DH	EH	FH
Mode Register Name	Clock mode register	INT1 mode register	Port 6 mode register	Shift mode register

If addr4 = 1-7, the instruction outputs the contents the accumulator to the port specified by 4 bit immediate data addr4.

When addr4 = 8-BH, the contents of the accumulator are output to the expansion port of 8243 I/O expander specified by addr4 via port 1.

When addr4 = CH-FH, the contents of the accumulator are transferred to the mode register specified by addr4.

OP3 (Output Port 3)

1 Instruction code : 0 1 1 1 0 0 1 1
2 Bytes = 1
3 Cycles = 1
4 Function = Port 3 - A

This instruction outputs the contents of the accumulator to the port 3.

OP54 (Output Port 5 and Port 4)

1 Instruction code = 0 0 1 1 1 1 1 1 | 0 0 1 1 1 1 0 0
2 Bytes = 2
3 Cycles = 2
4 Function = Port 5 - A, Port 4 - (HL)

This instruction outputs the contents of the accumulator to the port5 and the contents of the data memory addressed by the HL register to the port 4.

ANP addr6,n4 (And Port with immediate) .... SET A Only

1 Instruction code = 0 1 0 0 1 1 0 0 $I_3I_2I1I_oD_3D_2D_1D_o$

2 Bytes = 2

3 Cycles = 2

4 Function = Port (addr6) - Port (addr6) $\wedge$ n4

addr6=D_{3-0} : 2-BH, n4=I_{3-0} : 0-FH

addr6	2	3	4	5	6	7	8	9	AH	BH
Port Name	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7	8243 Port 4	8243 Port 5	8243 Port 6	8243 Port 7

When addr6 = 2-7, this instruction causes the contents of the port specified by 4 bits immediate data (addr6) to be ANDed with 4 bits immediate data (n4), and the result is output to the port specified by addr6.

When addr6 = 8-BH, the instruction ANDs the expansion port of 8243 I/O expander to be specified by addr6 with n4, and the result is output to the expansion port specified by addr6. These are interfaced with 8243 via port 1.

ORP addr4, n4 (Or Port with Immediate) .... SET A Only

1 Instruction code = 0 1 0 0 1 1 0 1 $I_3I_2I_1I_oD_2D_2D_1D_o$

2 Bytes = 2

3 Cycles = 2

4 Function = Port (addr6) - Port (addr6) $\vee$ n4

addr6=D_{3-0} : 2-BH, n4=I_{3-0} : 0-FH

addr6	2	3	4	5	6	7	8	9	AH	BH
Port Name	Port 2	Port 3	Port 4	Port 5	Port 6	Port 7	8243 Port 4	8243 Port 5	8243 Port 6	8243 Port 7

When addr6 = 2-7, this instruction causes the contents of the port specified by 4 bits immediate data (addr6) to be ORed with 4 bits immediate data (n4), and the result is output to the port specified by addr6.

When addr6 = 8-BH, this instruction causes the expansion port of 8243 I/O expander specified by addr6 to be ORed with n4 and the result is output to the expansion port specified by addr6 via port 1.

RPBL (Reset Port Bit specified by L)

1 Instruction code = 0 1 0 1 1 1 0 0

2 Bytes = 1

3 Cycles = 1

4 Function = 8243 Port bit (L) - 0

L_3L_2 \ L_1L_0	00	01	10	11
00	P40	P41	P42	P43
01	P50	P51	P52	P53
10	P60	P61	P62	P63
11	P70	P71	P72	P73

Port of 8243 I/O expander, bit specifying

This instruction resets (0) the bit designated by low-order 2 bits (L_1, L_0) of L register of the port (expansion port of 8243 I/O expander) specified by high-order 2 bits (L_3, L_2) of L register. It is interfaced with 8243 via port 1.

5 Example : When resetting bit 2 of port 6 of 8243.

```
    :
LLI  0AH  ;L=1010
RPBL      ;8243 P62←0
    :
```

SPBL (Set Port Bit specified by L)

1 Instruction code = 0 1 0 1 1 1 0 1
2 Bytes = 1
3 Cycles = 1
4 Function = 8243 Port bit (L) - 1

L_3L_2 \ L_1L_0	00	01	10	11
00	P40	P41	P42	P43
01	P50	P51	P52	P53
10	P60	P61	P62	P63
11	P70	P71	P72	P73

Port of 8243 I/O expander, bit specifying

This instruction sets (1) the bit specified by low-order 2 bits (L_1L_0) of the L register, of the port specified by high-order 2 bits (L_3, L_2) of the L register in the expansion port of 8243 I/O expander.

5 Example : When setting bit 3 of port 7 of 8243.

```
    :
LLI  0FH  ; L=1111
SPBL      ; 8243 P73←1
    :
```

12.5.14 CPU Control Instructions

HALT (Halt)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 0 1 1 0]

2 Bytes = 2

3 Cycles = 2

4 Function =

This instruction sets the halt mode

STOP (Stop)

1 Instruction code = [0 0 1 1 1 1 1 1] [0 0 1 1 0 1 1 1]

2 Bytes = 2

3 Cycles = 2

4 Function =

This instruction sets the stop mode.

NOP (No Operation)

1 Instruction code = [0 0 0 0 0 0 0 0]

2 Bytes = 1

3 Cycles = 1

4 Function =

This instruction spends one machine cycle without doing anything

ニーモニック	ページ	ニーモニック	ページ	ニーモニック	ページ
ACSC		LAMT		SKDEI	
ADSC		LAMTL		SKEEI	
AESC		LDEI		SKHEI	
AHSC		LDI		SKI	
AISC		LEI		SKLEI	
ALSC		LHI		SKMBF	
ANL		LHLI		SKMBT	
ANP		LHLT		SKMEI	
ASC		LLI		SLSB	
CALL		NOP		SMB	
CALT		OP		SPBL	
CMA		OP3		ST	
DDE		OP54		STOP	
DDRS		OPL		TAD	
DES		ORL		TAE	
DHL		ORP		TAH	
DI		POPDE		TAL	
DLS		POPHL		TAMMOD	
EI		PSHDL		TAMSIO	
EXL		PSHHL		TAMSP	
HALT		RAL		TCNTAM	
IDE		RAR		TDA	
IDRS		RC		TEA	
IES		RMB		THA	
IHL		RPBL		TIMER	
ILS		RT		TLA	
IP		RTPSW		TSIOAM	
IP1		RTS		TSPAM	
IP54		SC		XAD	
IPL		SDSB		XADR	
JAM		SESB		XAE	
JCP		SHSB		XAH	
JMP		SIO		XAL	
JMPL		SKABT		XAM	
LADR		SKAEI		XHDR	
LAI		SKAEM		XLDR	
LAM		SKC			

ニーモニック	ページ	ニーモニック	ページ	ニーモニック	ページ
ACSC		LAMT		SKLEI	
AISC		LAMTL		SKMBF	
ANL		LHI		SKMBT	
ASC		LHLI		SKMEI	
CAL		NOP		SMB	
CALL		OP		SPBL	
CMA		OP3		ST	
DDRS		OP54		STII	
DLS		OPL		STOP	
EXL		ORL		TAMMOD	
HALT		RAL		TAMSIO	
IDRS		RAR		TAMSP	
ILS		RC		TCNTAM	
IP		RMB		TIMER	
IP1		RPBL		TSIOAM	
IP54		RT		TSPAM	
IPL		RTS		XADR	
JAM		SC		XAH	
JCP		SIO		XAL	
JMP		SKABT		XAM	
JMPL		SKAEI		XHDR	
LADR		SKAEM		XLDR	
LAI		SKC			
LAM		SKI			

NEC cannot assume any responsibility for any circuits shown or represent that they are free from patent infringement.
NEC reserves the right to make changes any time without notice.

NEC

NEC Electronics (Europe) GmbH, Oberrather Str. 4, 4000 Düsseldorf 30, W.-Germany, Tel. (0211) 650301, Telex 8587419

NEC Electronics (Germany) GmbH, Oberrather Str. 4, 4000 Düsseldorf 30, Tel. (0211) 650302, Telex 8587419
- Hindenburgstr. 28/29, 3000 Hannover 1, Tel. (0511) 881013–16, Telex 9230109
- Bayerstr. 21, 8000 München 2, Tel. (089) 591364–68, Telex 522971
- Solitudestr. 218, 7000 Stuttgart 31, Tel. (0711) 881119, Telex 7252220
- NEC Electronics (BNL) – 33 Alard du Hamelstraat, 5622 CC Eindhoven, Tel. (040) 445855, Telex 51923
- NEC Electronics (Scandinavia) – Box 4039, S-18304 Täby, Tel. (08) 756 72 45, Telex 13839

NEC Electronics France, Tour Chenonceaux, 204, Rond Point du Pont de Sèvres, F-92516 Boulogne Billancourt, Tel. (01) 620-6400, Telex 203544

NEC Electronics Italiana S.R.L., Via Cardano 3, I-20124 Milano, Tel. (02) 67 09 108, Telex 315355

NEC Electronics UK, Limited, 116, Stevenston Street, New Stevenston, Motherwell ML 14LT, GB, Tel. (0698) 732221, Telex 777565